

A Unified Mathematical Language for Medicine and Science

Patrick St-Amant

Department of Mathematics,
Collégial International Sainte-Anne,
Montreal, Canada

Abstract

A unified mathematical language for medicine and science will be presented. Using this language, models for DNA replication, protein synthesis, chemical reactions, neurons and a cardiac cycle of a heart have been built. Models for Turing machines, cellular automaton, fractals and physical systems are also represented with the use of this language. Interestingly, the language comes with a way to represent probability theory concepts and also programming statements. With this language, questions and processes in medicine can be represented as systems of equations; and solutions to these equations are viewed as treatments or previously unknown processes. This language can serve as the framework for the creation of a large interactive open-access scientific database that allows extensive mathematical medicine computations. It can also serve as a basis for exploring ideas related to what could be called metascience.

Contents

Contents	2
1 A Vision of Medicine	6
2 Purpose of the Language	7
3 Essence of the Language	8
4 The Mathematical Language	11
4.1 Visual Language	11
4.2 Basic Notation	13
4.2.1 Forms	13
4.2.2 Transformations	13
4.2.3 Reduction	14
4.2.4 Series of transformations	15
4.2.5 Parallel transformations	17
4.2.6 Arising and Dissolution	19
4.2.7 Position Pairing in Transformations	19
4.2.8 Preserving Connections	21
4.2.9 Colors for Pairing	23
4.2.10 Distance Preservation	27
4.2.11 Transformation exceptions	28
4.3 Naming	28
4.3.1 Identifiers	28
4.3.2 Function Forms	29
4.3.3 Abstraction	31
4.4 Reduction view	34
4.5 Three dimensional language	34
4.6 Diagram versus Inline Notation	37
4.7 Equations and solutions	37
4.8 Finer and coarser models	39
4.9 Higher Order	39
5 Probabilities	42
6 Summary of the notations	45

7	Mathematical models in medicine	47
7.1	Cell-division cycle	48
7.2	Interphase	48
7.3	DNA replication	49
7.3.1	Diagram for the replication of the right leading strand	50
7.3.2	Diagram for the replication of the right lagging strand	51
7.3.3	Inline replication of the right leading strand	52
7.3.4	Inline replication of the left leading strand	52
7.3.5	Inline replication of the right lagging strand	53
7.3.6	Inline replication of the left lagging strand	54
7.4	Messenger RNA	54
7.5	RNA Splicing	55
7.6	Proteins	56
7.7	Heart model	57
7.8	Skin Healing	58
7.9	Neurons	59
7.9.1	Diagram model	60
7.9.2	Inline model	61
7.10	Biology diagrams	63
7.11	3D Mesh transformations	64
7.12	Meta-medicine	65
8	Chemistry	65
8.1	Chemical reactions	65
8.2	Studying hydrogen combustion	66
8.3	Reaction mechanisms	68
9	Absolute versus relative transformations	68
9.1	Relative molecules	68
9.2	Chains of transformations	69
10	Computing treatments and processes	70
10.1	Solving for processes	70
10.2	Solving for antiviral drugs	72
10.3	Understanding cancer	73
10.4	Calculating mathematical solutions	74
10.5	Producing the remedies	74

11 Computing techniques	75
11.1 Solving equations	75
11.2 Principle of resilience and propagation	75
11.3 Optimization	76
11.4 Main algorithm	77
11.5 Computable and uncomputable functions	77
12 Computer Science	78
12.1 Programming statements	78
12.1.1 If-statement	78
12.1.2 Switch-statement	78
12.1.3 For-loop	79
12.1.4 While-loop	79
12.1.5 Do-loop	80
12.2 Forms programming	80
12.3 Conway's game of life	81
12.4 Turing machines	82
13 Mathematics	82
13.1 Functions	82
13.2 Fractals	83
13.3 Differential calculus	85
13.4 Meta-mathematics	85
13.5 Continuous transformations	86
13.6 Formal description of the language	86
14 Physics	87
14.1 Motion	87
14.2 Attractive and repulsive forces	88
14.3 Future Models	90
15 Metascience	91
16 Appendix	93
16.1 Open Transformations	93
16.2 Negation	94
16.3 Initial set displacement	94
16.4 Series and parallel invariance	95

16.5 Typesetting	95
References	98

1 A Vision of Medicine

Imagine a world where treatments are computed and where knowledge of medicine is merged in a large database. New treatments are computed based on mathematical models built by researchers and each new model, like a piece of a puzzle, can be connected within a vast system of knowledge. Potential new treatments would be tested within a computational model and effects would be observed virtually, making animal experimentation obsolete due to lack of accuracy. New processes explaining functions of the body would be computationally deduced and hypothesized. Ultimately, treatments with minimal sides effects would be tailored to each patient based on their personal biological data.

It is difficult to keep up-to-date with new important research and publications and impossible to cover all the new daily literature in one's domain of study. We are at a point where even a group of experts with the best communication tools and habits cannot efficiently integrate all the new knowledge of their field. In contrast, we now have the technological means to gather and analyse large amount of data.

A key component needed for the realization of this future is a common language for science and medicine. In this paper, we present a unified mathematical language which can be used to represent biological processes and scientific concepts. Although the examples are geared towards medicine, we will also present how this language can be used in different domains with the aim of demonstrating that effort in developing such a language can lead to improved communication and exchange of ideas between fields of studies.

Translating each research paper into a set of mathematical expressions and integrating it into a large database would ensure the findings of each new research study would be integrated into the existing body of knowledge. A feature of this language is that it could encompass the results of a vast number of papers, so that computations, discoveries and analysis could also be done on the field of medicine as a whole field. This could be a step into a new domain which could be called meta-medicine. This could be an opportunity to bridge the gap between publishing results and the integration of the results by the community. Competing theories and contradicting results could be tested against each other through computation and help to improve the general model. Interestingly, experimental results which are usually considered inconclusive or 'negative' could also contribute to enrich the global database.

2 Purpose of the Language

The objective behind these investigations is to build a language in which all scientific concepts and notations can be represented, while keeping the way each discipline functions and by enhancing the computational nature of each discipline. Along with the ability to represent all scientific concepts and its computational power, we would also need a language that allows navigation between different levels of complexity or scales (from DNA replication to body mechanics), can be used in a large database, has a visual notation and compact notation and has a relative ease of use regardless of the domain of study. With this language, dynamic models in biology, chemistry, neuroscience, computer science and physics have been built. We hope that sufficient steps have been taken towards demonstrating that the language has the qualities mentioned above and that this will inspire others to explore it further.

In mathematics, set theory combined with mathematical logic is known to be able to represent most mathematical expressions and concepts, but in practice we rarely directly use set theoretical expressions to represent or solve mathematical problems. Similarly, in computer science, although all programs can be reduced to Turing machines of zeros and ones, we rarely think or interact with high-level programming languages in this way. Whatever the language, there seems to be a gap between what we do in the discipline and the fundamental language. We could say that English along with all mathematical symbols is a complete language which can communicate any scientific concept, but unfortunately, computations cannot be done efficiently on words and sentences. The mathematical languages known as rewriting systems [1] are very powerful and have been shown to represent grammar, plants and fractals [2, 5]. However, the scope of these systems is sometimes restricted to certain types of objects and focuses on automatic theorem proving, normal forms (expressions that cannot be transformed further) and termination.

The language which will be presented can be viewed as a higher order rewriting language that has no restrictions on the type of objects and on the objects that can be named. Alternatively, the language could be viewed as a higher order universal Turing machine where the symbols, states and tape can be any type of object. Furthermore, we could say that the head of the Turing machine could affect its own states and the symbols on the tape could also affect the states and properties of the head. Example of types of objects are models of atoms, molecules, DNA strands, cells, neurons, con-

cepts, mathematical equations, programs, 3D geometrical structures, people and also infinite collections of different types of objects.

The driving question underlying this study can be written as:

What are the atoms of scientific concepts and how can we combine them to represent complex systems?

3 Essence of the Language

In this section, we introduce the elementary elements of the language. This is only meant to offer a quick glance at the language and one should read further in the subsequent sections for a more accessible and progressive description of the language along with key applications to biology, medicine and other scientific domains.

Everywhere we look in the world, we see change. We observe changes at every level, whether it be atoms, humans or the stars; they are changing all the time. As observers, we can take note of these changes and record them. Based on these records, we can infer and extrapolate the past and the future. In studying these records, we recognize interesting structures and name them. The principles behind the unified mathematical language that will be presented can be reduced to the following two statements:

1. Observe a change in the world. Record what it was and what it became.
2. All objects and changes can be named.

For example, let A and B be objects. We will denote the transformation of object A into B by the notation

$$\boxed{B \triangleright A}.$$

This notation should be understood as B having the potential to slide down over the triangle \triangleright in the direction of A so that B will take the place of A . In itself, this object that we call a *transformation* is interesting, but it only represents a potential. For the change to occur, we first need the object A to be present. This will be denoted by

$$\boxed{B \triangleright A} \rightarrow (A).$$

The right arrow means that we apply the transformation $\boxed{B \triangleright A}$ to the object A in the ordered set (A) . Therefore, changing A into B will result in (B) . In other words, A is changed into B by applying the transformation $\boxed{B \triangleright A}$. This notation should be visualized as though the transformation will slide down the arrow \rightarrow until the A on the right-hand side of the transformation superposes over the A inside the parentheses. When the A 's are matched, the B slides down over the directed triangle to finally replace A .

Mathematically, the application of a transformation can be written this way by using the double arrow as follows.

$$\boxed{B \triangleright A} \rightarrow (A) \Rightarrow (B)$$

This reads as the system which is composed of the transformation $\boxed{B \triangleright A}$ applied to the object (A) which *reduces* to the object (B) .

When we hear the name of friend, we can access a lot of knowledge concerning our friend. The short string of letters of the name points to an array of knowledge and permits us to avoid the lengthy description of all that is known about the friend. Naming objects is also very important in mathematics and science. Scientific terms permit us to quickly build new theories and explain insightful understanding about the world. The calculative power of mathematics can be said to rely on its ability to condense long expressions and operations into a string of symbols where the operations are viewed as names that indicate how the symbols should be acted upon. In our language, we will name objects and transformations by using the symbol $:=$. For example, we can give the name t to our transformation $\boxed{B \triangleright A}$ by writing $t := \boxed{B \triangleright A}$.

For this language to be powerful, a key is to allow A and B to be anything we want. For example, we can take A and B to be numbers, models of DNA strands, neurons, people, behaviours, 3 dimensional geometric objects or even groups containing different types of objects.

Based on this approach, this means that the action of folding a model of a protein P into the structure S is written as $FoldingProtein := \boxed{S \triangleright P}$ and representing a person closing his hand can also be written in the same language as $ClosingHand := \boxed{\text{closed hand} \triangleright \text{open hand}}$. A physical therapist will usually be interested in body movements while the biologist will be concerned about objects of the size of proteins, but now both can use this language without losing their respective level of interest. Thus, everything can be brought back to this language and facilitate comparisons and

knowledge transfer between disciplines.

This way of viewing things can seem overly simplistic when we think of all the numerous systems and concepts in science it needs to represent, but this is similar to explaining all the physical objects we observe with only the concept of atoms. We will see that the combinations of transformations leads to highly complex systems such as the functioning of the body. As mentioned, transformations can act on any type of object, but they can also be applied in sequence or in parallel to groups of objects. Moreover, transformations can be applied to groups of transformations so that we can describe and study higher order languages. Also, it is possible to have transformations which apply to themselves, thus opening a wealth of possibilities.

Importantly, the language comes with computational power because a programming statement can be viewed as a collection of transformations of different types of objects. Moreover, we will see that scientific questions can be written as systems of equations such that the solutions to these equations are the answers. This way, each domain can rely on computing tools to deepen their respective explorations. In medicine, this means that with some effort in translating, creating models and computing, we can build a large dynamic model of the body where all different levels are represented. We can also insert variables into our models and create equations that represent important questions in medicine. Solutions to these equations would represent treatments or unknown biological processes, thus allowing us to concentrate our resources on solving certain equations to unlock new types of treatments or understanding.

Using transformations, which can be seen as the atoms of scientific concepts, we will be able to represent many systems. For example, DNA replication can be seen as composed of transformations aimed at separating the strands and transformations replacing nucleotides by pairs of nucleotides. Turing machines can be viewed as a collection of transformations which change series of zeros and ones on a tape. A mass falling to the ground can be represented by a transformation which replaces the mass at a certain height by the same mass slightly closer to the ground. We will present more refined versions of these examples in the following text and also give models for messenger RNA, proteins, a simplified heart, neurons, cell division, chemical reactions, programming statements, mathematical functions and cellular automata.

Another important outcome of this language is that discoveries in one scientific domain can lead to similar observations in another domain as the

domains share similar mathematical structures. This points to what could be called *metascience*. Having a seamless language for computer science, mathematics and biology would permit us to transform abstract mathematical concepts into the field of biology, and conversely, study abstract biological processes as mathematical structures. We could say that a large part of modern mathematics stems from the plane geometric figures studied by the ancient Greeks, fractions and numbers. In light of this, one might wonder which kind of mathematical structures will be studied in the future.

4 The Mathematical Language

In the following section we will describe the language, present important types of transformations, define the equations of this language and discuss some properties of the language.

4.1 Visual Language

When we visualize molecules; we usually imagine them as a drawing or a three dimensional structure. One of the most efficient ways to learn is by seeing someone do it, the next best thing for many people is to see a drawing or an illustration for which there is a textual description. The inline notation where all is written as text is valuable if we want a compact form of a concept, but can be more difficult to understand. For example, molecules are more accurately represented in three dimensions, but they have a two dimension representation and an inline notation. If we do not want to lose information while reducing the number of dimensions, we have to add new symbols and notations making the representation less easily understood.

For example, a D-glucose chain can be represented inline as $C_6H_{12}O_6$ while the 2D skeletal notation and the 3D representation give more information as seen below. Note that, while not easy to read, it is possible to encode inline the 3D representation of a molecule.

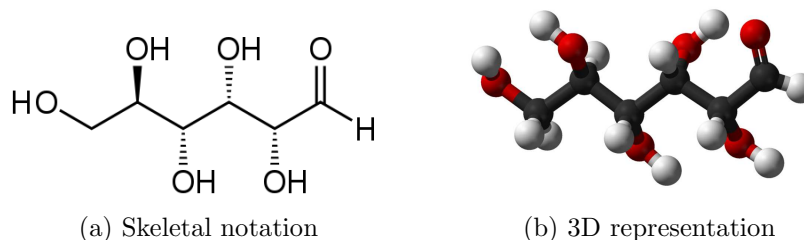


Figure 1: D-glucose chain

Usually, a mathematical language is restricted to certain types of symbols such as numbers, letters or operators. To have an all-encompassing language, we have to extend our notion of what a symbol is. For us, we will allow the symbols to be graphs, diagrams, geometrical shapes and even 3D objects (represented in 2D or not). This allows us to have a language which has a two dimensional notation, an inline notation and a three dimensional version. Diagrammatics notation is already being successfully used in physics, mathematics and biology. In Physics, Feynmann diagrams [3, 4] have proven to be a powerful tool and Penrose’s tensor diagrammatic notation is proposing a new way to do tensor calculus [7]. In mathematics, diagrams are at the heart of graph theory and knot theory. As a notation, the language of commutative diagrams of category theory is now an essential tool in investigating highly abstract mathematical structures [6] and different diagrammatic notations have been used to further study category theory [8]. In Biology, textbooks use many types of diagrams to help the reader to understand. While these diagrams are not standardized, it could be fruitful to have a standard notation to enhance access to knowledge. With all the new capabilities of text editing and access 3D software, we are in a position to transition out of line-by-line mathematical notation and explore diagrams and 3D notations.

In this paper, to help understand, we will often use the two dimensional visual language and make extensive use of colors to simplify the notation. Colors can be always be replaced by numbered subscript or superscript, but its use helps to make the notation lighter. The visual notation is good for communication and understanding, but the inline notation is very important since it is the basis of computation. It will quickly become clear that the inline version of a structure can be cumbersome, since even colors need to be replaced by a subscript or superscript notation. We will introduce the visual notation in parallel with the basic inline notation. In the construction

of mathematical models, we will not restrict ourselves to a particular one, but will switch between them in a way which is beneficial for understanding.

4.2 Basic Notation

4.2.1 Forms

In our language, any collection of symbols will be called a *form*. There are no restrictions on what a form can be. They can be letters, numbers, mathematical symbols, graphs, planar figures, 3D geometric shapes, people and collections of objects. We will only have to be careful when handling the special characters ‘ \triangleright ’ and ‘ $=$ ’. This language could be seen as a calculus of forms where forms can be anything we decide to name or distinguish.

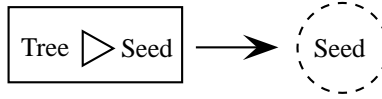
4.2.2 Transformations

As time passes, things or the forms we see are always changing. A seed becomes a sprout and the sprout becomes a tree. In our attempt to understand the world, we make observations such as “this type of seed will give a certain type of tree with many different characteristics such as the shape of the leaves and cellular structure”. In the case of the seed-tree observations, we usually think of the seed as being the cause of the tree and not the other way around. As presented above, we will denote this as $\boxed{Tree \triangleright Seed}$ and call such a structure a *transformation*. Transformations are forms having potential to change other forms. The transformation $\boxed{Tree \triangleright Seed}$ can be understood as representing the process where a Seed is replaced by a Tree, a seed transforming into a tree or a seed becoming a tree. The right-hand side of the symbol \triangleright will be called the *cause* of the transformation and the left-hand side will be called the *effect* of the transformation. It should be visualized as the potential for the word *Tree* to slide down the directed triangle and replace the word *Seed*.

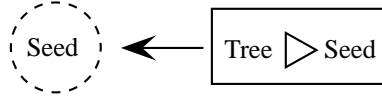
To observe the process of the becoming of a tree from a seed; there needs to be a seed present. This can be represented as $\boxed{Tree \triangleright Seed} \rightarrow (Seed)$ and will be called a *system*. The Seed at the right end of the arrow and between the parentheses ‘ $()$ ’ indicates the presence of the Seed. This ordered set that is acted on by some transformations will be called the *initial* form of the of these transformations. In the present case, we say that $(Seed)$ is the initial form of the transformation $\boxed{Tree \triangleright Seed}$. The structure $\boxed{Tree \triangleright Seed}$ can

be seen as the potential of transformation of a seed into a tree and will only transform a seed in a tree when a seed is present. Eventually, we will have many elements in the initial form and unless said otherwise, we will consider that order in which the element appears as important. For example, (A, B) is not considered to be the same as (B, A) .

We can represent the system $\boxed{Tree \triangleright Seed} \rightarrow (Seed)$ in a diagram form as follows.



Or it can be represented as the following diagram, since it is understood that the side to where the triangle is pointing is the form that will be replaced.



In the diagram notation, the initial form will be denoted by a shape drawn with dash lines. We will use mostly shapes such as dashed circles, ellipses and rectangles.

4.2.3 Reduction

The act of applying the transformation $\boxed{Tree \triangleright Seed}$ to the form $(Seed)$ is said to be a *reduction* of the system $\boxed{Tree \triangleright Seed} \rightarrow (Seed)$ to the form $(Tree)$. This reduction is indicated by a double arrow ' \Rightarrow ' and is written as

$$\boxed{Tree \triangleright Seed} \rightarrow (Seed) \Rightarrow (Tree).$$

This expression is read as the transformation $\boxed{Tree \triangleright Seed}$ applied to the form $(Seed)$ and reduces to the form $(Tree)$. It is important to note that the transformation is applied only once and then disappears after the reduction. When multiple transformations are used in a system, a reduction can refer to the final form where all transformations have been applied or all the steps leading to that final form are complete.

Seeds come in many types, so we can extend our language further and apply the transformation such as $\boxed{Pine \triangleright Seed}$ to multiple seeds. There are two interesting ways to do this. The first is by successively applying certain

transformations to an ordered set containing multiple seeds. The second is by applying one transformation at a time without any definite order. The first is said to be applied in *series* and the second in *parallel*.

The number of transformations which can be applied on an initial form can be finite or infinite. In the theory of term rewriting systems, a main focus is to have systems which terminate after a finite number of reductions. For us, since we are mainly concerned about creating mathematical models, we are interested in all the steps of the reductions and this whether it terminates or not. An interesting question for us to ask is if a system has a period or a cycle. In biology, an example of periodic system is the cell cycle.

4.2.4 Series of transformations

A series of three transformation applied to an ordered set of three seeds is written as

$$\boxed{Maple \triangleright Seed} \rightarrow \boxed{Oak \triangleright Seed} \rightarrow \boxed{Pine \triangleright Seed} \rightarrow (Seed, Seed, Seed)$$

The reduction sequence of this system is be done in three steps and is denoted as follows. The transformation closest to the initial form is applied first followed by the second closest and so on.

$$\begin{array}{c} \boxed{Maple \triangleright Seed} \rightarrow \boxed{Oak \triangleright Seed} \rightarrow \boxed{Pine \triangleright Seed} \rightarrow (Seed, Seed, Seed) \\ \downarrow \\ \boxed{Maple \triangleright Seed} \rightarrow \boxed{Oak \triangleright Seed} \rightarrow (Seed, Pine, Seed) \\ \downarrow \\ \boxed{Maple \triangleright Seed} \rightarrow (Oak, Pine, Seed) \\ \downarrow \\ (Oak, Pine, Maple) \end{array}$$

Since all the elements of the initial form are the same, the transformation $\boxed{Pine \triangleright Seed}$ can replace any choice of Seed. If one wants to differentiate between the Seeds, we could rename them as in the following example.

$$\boxed{Maple \triangleright Seed3} \rightarrow \boxed{Oak \triangleright Seed2} \rightarrow \boxed{Pine \triangleright Seed1} \rightarrow (Seed3, Seed1, Seed2)$$

We now introduce superscript to the transformations. If we have a series of the same transformation being applied on an initial form, we will write this in a more compact notation by writing as a superscript an integer followed by the letter ‘*S*’ which indicates that the transformation is applied in series. For example,

$$\boxed{Tree \triangleright Seed} \rightarrow \boxed{Tree \triangleright Seed} \rightarrow \boxed{Tree \triangleright Seed} \rightarrow (Seed, Seed, Seed)$$

can be written as

$$\boxed{Tree \triangleright Seed}^{3S} \rightarrow (Seed, Seed, Seed).$$

Another example with two different types of transformation is to write

$$\boxed{Oak \triangleright Seed}^{7S} \rightarrow \boxed{Pine \triangleright Seed}^{10S} \rightarrow (Seed, Seed, Seed, Seed, Seed)$$

to represent 10 successive applications of the transformation $\boxed{Pine \triangleright Seed}$ followed by a series of 7 applications of $\boxed{Oak \triangleright Seed}$.

It is important to note that when no superscript is written, it is assumed to mean a superscript of 1*S*. For example, $\boxed{Tree \triangleright Seed}^{1S}$ is equivalent to $\boxed{Tree \triangleright Seed}$.

We can also use the infinity symbol ‘ ∞ ’ to denote that the transformation is continuously applied when needed. Thus the system

$$\boxed{Oak \triangleright Seed}^{\infty S} \rightarrow (Seed, Seed, Seed, Seed, Seed),$$

reduces to

$$\boxed{Oak \triangleright Seed}^{\infty S} \rightarrow (Oak, Oak, Oak, Oak, Oak).$$

If we want to replace all that can be replaced, we use the sharp symbol ‘ \sharp ’ to denote that the transformation is applied until it reaches a step where it cannot replace another element of the initial form, then the transformation disappears. In other words, it replaces all it can and then disappears. For example, the system

$$\boxed{Oak \triangleright Seed}^{\sharp S} \rightarrow (Seed, Seed, Seed, Seed, Seed),$$

reduces to

$$(Oak, Oak, Oak, Oak, Oak).$$

4.2.5 Parallel transformations

When transformations are applied in parallel, we will denote this by grouping them in box brackets ‘[]’. For example, two transformations applied on an initial form containing three seeds is written as

$$[\boxed{Pine \triangleright Seed}, \boxed{Oak \triangleright Seed}] \rightarrow (Seed).$$

When parallel transformations are applied to an ordered set, any transformation that can be applied can be chosen to be applied first. After it has been applied, it disappears and another one is chosen. An example of reduction is

$$\begin{aligned} &[\boxed{Pine \triangleright Seed}, \boxed{Oak \triangleright Seed}] \rightarrow (Seed) \\ &\quad \Downarrow \\ &[\boxed{Oak \triangleright Seed}] \rightarrow (Pine) \end{aligned}$$

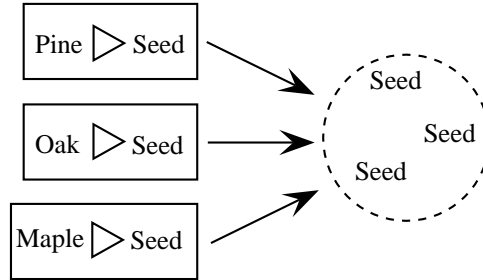
But since the transformations were in parallel, we could also have

$$\begin{aligned} &[\boxed{Pine \triangleright Seed}, \boxed{Oak \triangleright Seed}] \rightarrow (Seed) \\ &\quad \Downarrow \\ &[\boxed{Pine \triangleright Seed}] \rightarrow (Oak) \end{aligned}$$

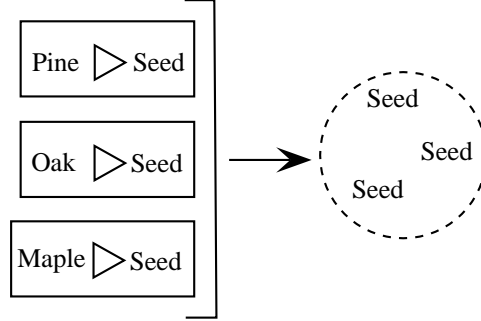
We now introduce the diagram notation. The system

$$[\boxed{Pine \triangleright Seed}, \boxed{Oak \triangleright Seed}, \boxed{Maple \triangleright Seed}] \rightarrow (Seed, Seed, Seed),$$

which contains three parallel transformations can also be represented as the following diagram.



It is also possible to reduce the number of arrows in the diagram by grouping the transformation under a bracket ‘[]’ as seen below.



Similarly to series of transformations, there is a superscript notation if the same transformation is applied multiple times on an initial form. This is done by writing as a superscript an integer followed by the letter ‘*P*’ which indicates that the transformation is applied in parallel. For example,

$$[\boxed{Tree \triangleright Seed}, \boxed{Tree \triangleright Seed}, \boxed{Tree \triangleright Seed}] \rightarrow (Seed, Seed, Seed)$$

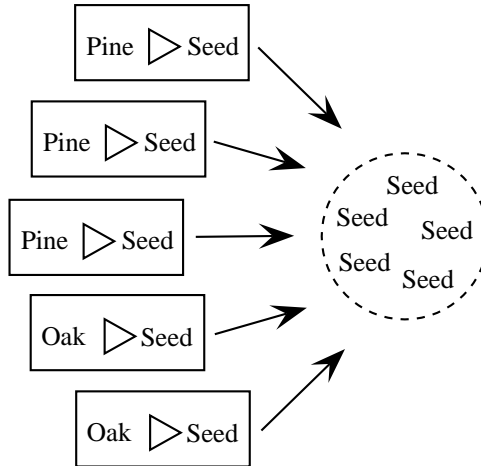
can be written as

$$[\boxed{Tree \triangleright Seed}]^{3P} \rightarrow (Seed, Seed, Seed).$$

Another example with two different types of transformations, is to write

$$[\boxed{Oak \triangleright Seed}]^{2P}, [\boxed{Pine \triangleright Seed}]^{3P} \rightarrow (Seed, Seed, Seed, Seed, Seed)$$

to represent the following diagram.



In a similar way to series of transformations, we can also use the infinity symbol ‘ ∞ ’ to denote that a transformation is continuously applied and the sharp symbol ‘ \sharp ’ to indicate that the transformation replaces all it can from the initial form and then disappears.

4.2.6 Arising and Dissolution

This language permits creation of new objects in an ordered set by using a transformation in which the right or left hand-side of the transformation is empty. An example is given by the following transformation of making rabbits appear in a grass and flower field. This is an interesting property, since this gives us a mathematical way to add new objects to a set of forms.

$$\begin{array}{c}
\boxed{\text{rabbit} \triangleright} \xrightarrow{3S} (grass, flowers) \\
\Downarrow \\
\boxed{\text{rabbit} \triangleright} \xrightarrow{2S} (grass, flowers, rabbit) \\
\Downarrow \\
\boxed{\text{rabbit} \triangleright} \xrightarrow{\quad} (grass, rabbit, flowers, rabbit) \\
\Downarrow \\
(grass, rabbit, flowers, rabbit, rabbit)
\end{array}$$

Similarly, as seen in the following example we can dissolve a form.

$$\begin{array}{c}
\boxed{\triangleright cup} \xrightarrow{2S} (cup, spoon, cup) \\
\Downarrow \\
\boxed{\triangleright cup} \xrightarrow{\quad} (cup, spoon) \\
\Downarrow \\
(spoon)
\end{array}$$

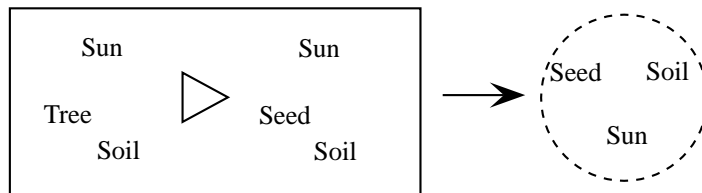
4.2.7 Position Pairing in Transformations

Until now in our transformations, there was only one term on each side of the directed triangle. We now allow multiple terms on each side of the triangle. Let’s revisit our seed example.

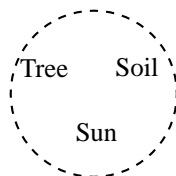
A seed turns into a tree if there is a soil, water and sun. After the appearance of the tree the soil is still there and the sun is still present. We could view the sun and the soil to be the supporting conditions. After a transformation is applied, the supporting conditions will stay there and will not change. In practice, we can say that a very small amount of soil and

energy from the sun was used, but in our present model, we will consider that it is a negligible amount. To do this we will identify together elements on each side of the triangle. This is done by respecting the position of the terms on both sides of the directed triangle. To indicate that a form A in a certain position has been replaced by another form B , we write the A and B at the same position on each side of the directed triangle. If a form is unaffected by the transformation, the form will appear at the same place on both sides. Another way to view this is to think that B has been paired with A . This is similar to functions where each element of the domain is paired with an element of the range of the function.

Here is an example of our seed turning into a tree with the conditions of sun and soil.

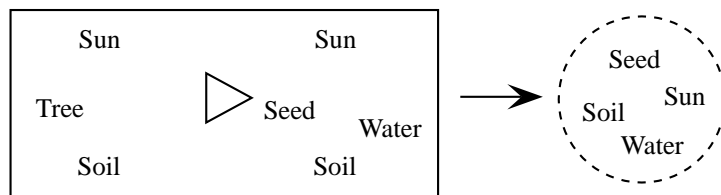


This will reduce to

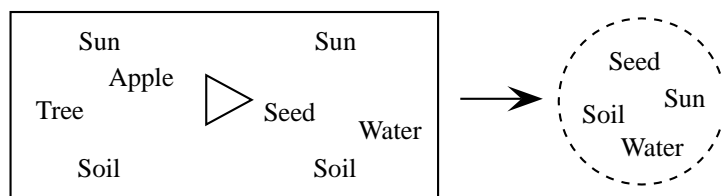


Notice that the terms of initial form are not in the same configuration as seen in the transformation. The position is only important in the transformation and is meant to indicate which forms are paired together. When a transformation is applied on the initial form, the elements are replaced at the place they are found in the initial form as seen above.

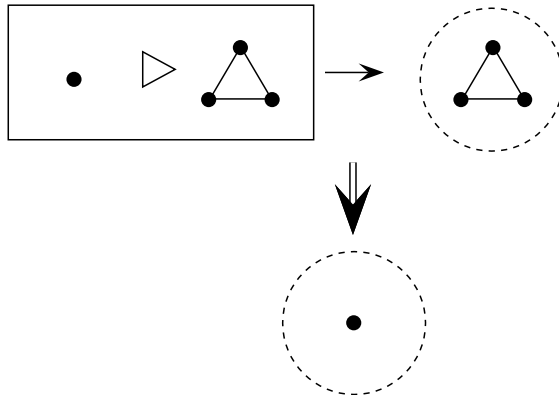
We now look at another example where a seed and water becomes a tree based on the conditions of the sun and soil. Notice that the water disappeared and that the seed was replaced by a tree. Note that this system makes an implicit use of the dissolution transformations when the water is removed.



In the next example, the water disappeared, the seed was replaced by a tree and an apple appears in some position.



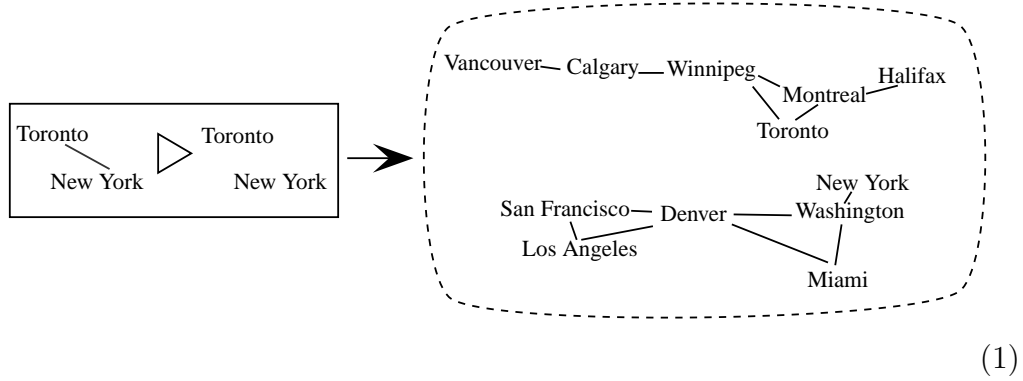
The next example shows that the replacing object can be positioned at another position than the replaced object. Here the triangle contracts to a point in the center of the triangle.



4.2.8 Preserving Connections

In science we are often interested in objects which have connections with one another. For example, atoms are connected by bonds to create molecules and networks of cities are sometimes represented as graphs of nodes connected by edges. We will later see that this will be very useful when we will be representing biological process such a DNA replication and representing a peptide bond.

Let's now look at a simplified network of the Canadian postal service with the cities of Vancouver, Calgary, Winnipeg, Montreal, Halifax and Toronto. In the United States, the simplified postal network covers the cities of New York, San Francisco, Denver, Washington, Los Angeles and Miami. After an agreement between both postal services; it is decided that both networks should be connected between the cities of Toronto and New York. Using our notation, this can be represented in the following transformation.



Most of the time, to represent connections between forms, we will use lines or directed arrows. When we need to modify a form connected to other forms, we will consider that transformations will preserve connections. More precisely, we say that:

If form A is replaced by form B , all the connections that node A has are transferred to B and B keeps its previous connections.

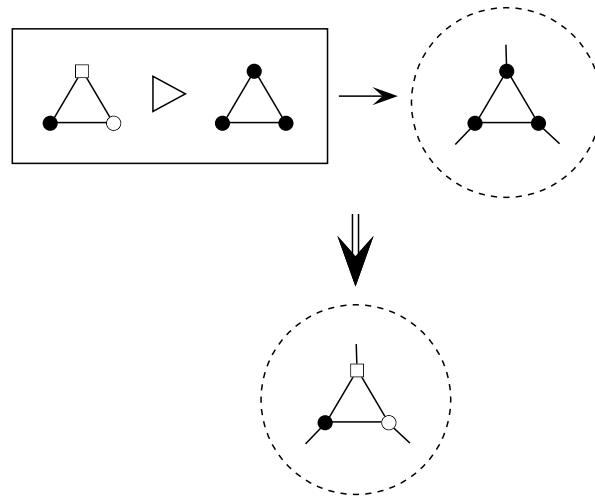
To illustrate this, we give the following two simple examples where $A - C$ is interpreted as node A is connected to node B .

$$\begin{array}{c} \boxed{B \triangleright A} \rightarrow (A - C) \\ \Downarrow \\ (B - C) \end{array}$$

and

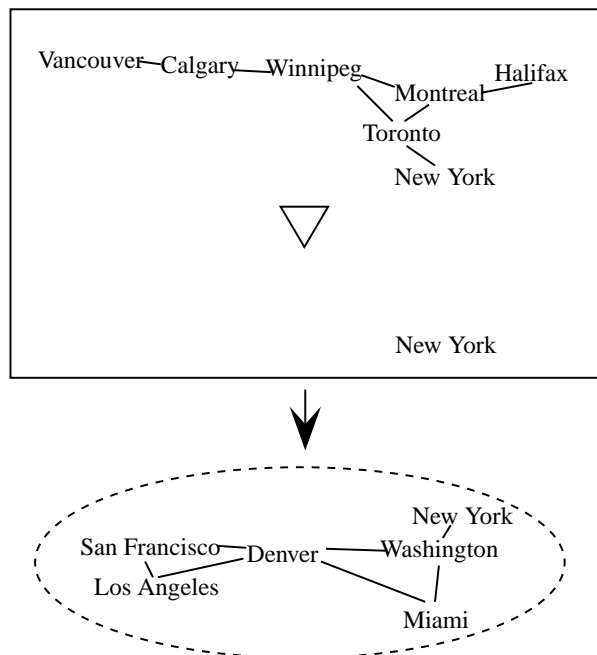
$$\begin{array}{c} \boxed{B \triangleright A} \rightarrow (D - A - C) \\ \Downarrow \\ (D - B - C) \end{array}$$

If we add connections to the contracting triangle we have seen above, we have the following reduction where the connections of the black circle nodes are transferred to the empty square and empty circle nodes.



4.2.9 Colors for Pairing

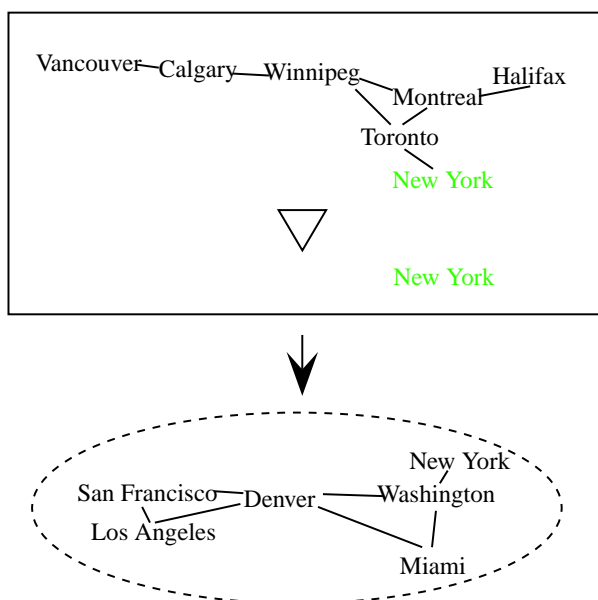
Returning to our example of the Canadian and US postal services, another way to connect Toronto and New York is to write the following.



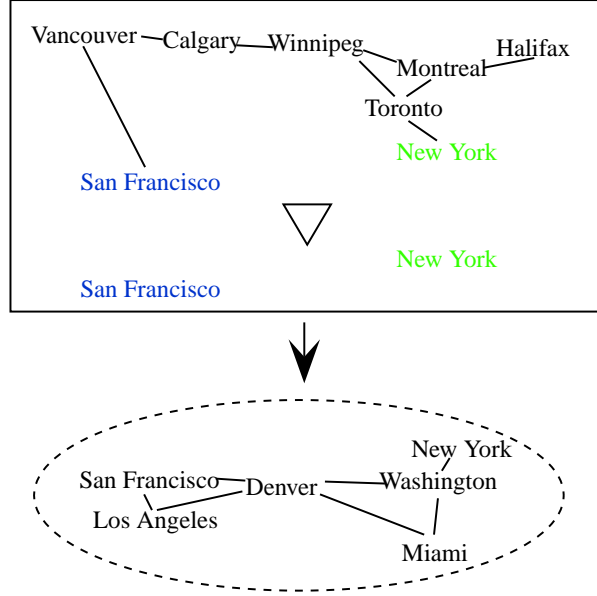
From this, it is not clear if the position of New York on the small side

of the directed triangle corresponds to New York or Montreal on the other side. Also, it is not a very compact notation. A solution to this problem is to introduce colors to help pairing forms. Forms of the same color on both sides of the directed triangle are paired together, meaning that the form being replaced is substituted by the form of the same color. Here we consider that black is not a color.

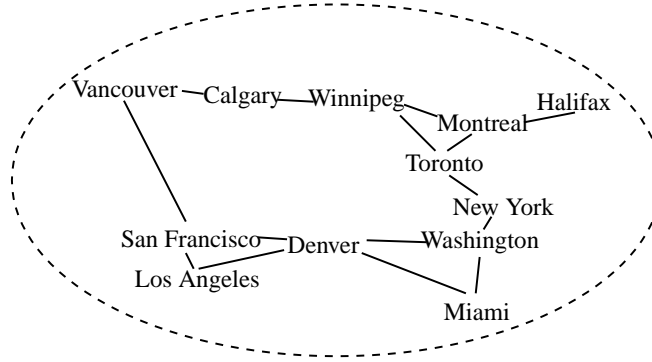
We can now rewrite our network while making sure that the connection is made with New York and not Montreal. The two postal networks are now linked together through Toronto and New York.



We can also use two colors to indicate the pairing. The blue color in the following example can be interpreted as connecting San Francisco with Vancouver and New York with Toronto.



This systems reduces to



Take the system $\boxed{D - B \triangleright A} \rightarrow (A - C)$ written inline where the connections between the nodes identified by letters, it is not clear how we should interpret it. There are a few possibilities. If we replace A by $D - B$, what will happen with the connection between A and C ? We could replace the form A by B and keep the connection between B and D or replace A by D and keep the connection between D and B . For this example, we can also use a color to indicate what should be paired together in the transformation.

By identifying A with B using the color green, we are saying that A is replaced by B and not by $D - B$. Thus, we have the following reduction.

$$\boxed{D - \textcolor{red}{B} \triangleright \textcolor{blue}{A}} \rightarrow (A - C)$$

$$\Downarrow$$

$$(D - B - C)$$

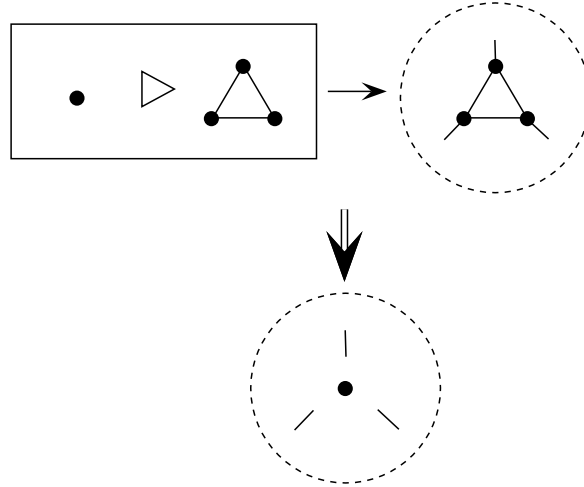
If we choose another color pairing, we find a different reduction.

$$\boxed{\textcolor{red}{D} - B \triangleright \textcolor{blue}{A}} \rightarrow (A - C)$$

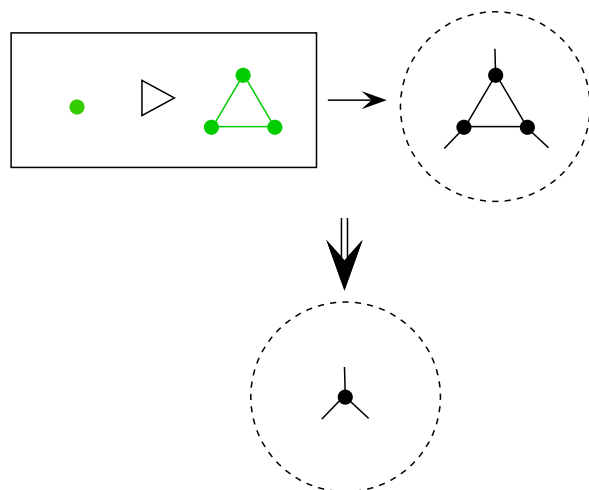
$$\Downarrow$$

$$(B - D - C)$$

Colors indicate which form should be replaced by which other form, but they also ensure that the connections are preserved. We have stated that if a form is in the same position, then the connections are preserved. But if the position of objects in a transformation does not correspond, how can we preserve the connections? For example, the contracting triangle with connections must be reduced as follows.



By using a color, we ensure that the connections are preserved since we are clearly indicating which objects are corresponding.



4.2.10 Distance Preservation

The distances between the elements in the initial form are important when we use a dashed line or the inline parentheses notation ‘()’. This is useful for ordered sets, for models of maps and objects in space like molecules. This means that terms of the same color can appear in different positions in a transformation and will move the corresponding element in the initial form.

If we represent the initial form with a full line or the inline parentheses notation ‘{ }’, this will mean that the position is not important. Even if the position is not important; the connections stay connected even if the nodes are moved around. This is used for sets where the order is not important and for graphs where the length of edges or position of the vertices are not relevant.

It is important to note that by default a transformation with disjoint elements on the right-hand side of a transformation does not require that the order or the space between them to be the same as in the initial form to be applied. For example,

$$\boxed{X, Y \triangleright C, A} \rightarrow (A, B, C)$$

reduces to (Y, B, X) .

When reading this transformation, we should view X, Y to be the effect of the transformation and C, A to be the cause. The commas and the position of these letters is important. It means that the transformation will

search the initial form for a C with the potential to replace it by X and will search the initial form for a A with the potential to replace it by Y , but the transformation will only be applied when C and A are present in the initial form.

If we want the order or the space to be respected, we will write *rigid* on the subscript of \triangleright . Thus,

$$\boxed{X, Y \triangleright_{\text{rigid}} C, A} \rightarrow (A, B, C)$$

does not reduce further, but the following system does.

$$\boxed{X, Y \triangleright_{\text{rigid}} B, C} \rightarrow (A, B, C) \Rightarrow (A, X, Y).$$

4.2.11 Transformation exceptions

A transformation which replaces a form A by a form B will replace A by B regardless of what is around A . If we want this transformation to not replace B when C is present with A in the initial form, we can write C in red to mean that the transformation will not replace A when C is also present in the initial form. Thus, the transformation $\boxed{B \triangleright A, C}$ will reduce the initial form (D, A, F) to (D, B, F) , but will leave (C, B, F) unaffected because C is present in the initial form.

We will consider the color red to be a reserved color and will only be used to indicate that a transformation cannot be applied if the form in red appears in the initial form.

4.3 Naming

We will now introduce three type of naming: the simple identifier or label, a type of naming which uses variables and a way to name based on abstraction.

4.3.1 Identifiers

We have already briefly introduced the *naming* symbol $:=$ to name transformations. In what follow, we will explain more concerning this notation and we will introduce a construction similar but more flexible than the objects and classes in object-oriented programming.

The reason for the $:=$ next to the equality symbols indicates that the symbols on the right is the name or label associated to the transformation written on the left. For example, $Growth := \boxed{Tree \triangleright Seed}$ or $\boxed{Tree \triangleright Seed} =:$

Growth means that we give the label *Growth* to the transformation $Tree \rightarrow Seed$. If we want to change or shorten the name of a transformation we do this by using the symbol $:=$. For example, we can make G_1 a perfect synonym of *Growth* by writing $G_1 := Growth$. We can also give names to initial sets. For example, $Field := (Seed, Seed, Seed, Soil)$. Such a name or label given to a form will be known as an *identifier*.

The understanding of this notation is similar to the usual understanding of mathematical equations. For example, if we have the equation $2x + y = 1$ and we know that $y = 4$ we can replace y by 4 in the equation to get $2x + 4 = 1$. We will treat the naming symbols in the same way. For example, if we have the system $\boxed{Tree \triangleright Seed} \rightarrow Field$ where $Field := (Seed, Seed, Seed, Soil)$, then we can write

$$\boxed{Tree \triangleright Seed}^{3S} \rightarrow (Seed, Seed, Seed, Soil).$$

Therefore, naming a form means that we can replace each occurrence of this form by this name and each occurrence of this name in a form can be replaced by the form it represents.

When reducing a system, it is important that we have objects of the same type in the initial form and in the transformations. For example, $\boxed{Tree \triangleright Seed} \rightarrow Field$ could not be reduced unless we replace the name *Field* so that we have the system

$$\boxed{Tree \triangleright Seed}^{3S} \rightarrow (Seed, Seed, Seed, Soil).$$

4.3.2 Function Forms

In object-oriented programming, classes are powerful tools that allow encapsulation of data, help making a program more transparent and encourage code reusability. One way to understand classes and objects is given in the following general example.

If we are writing a program that refers to different horses, we would need to access all these different horses. Every time we need to refer to a horse in our program, we could write the code describing this horse. If we have to refer to many different horses this could be quite long and repetitive. Another way to do this is to build a class or template for all types of horses. Suppose we are only concerned with characteristics such as color, age, running speed, height and tameness. We can define a class called ‘Horses’ that will group the attributes of color (C), age (A), running speed (S), and the behaviours

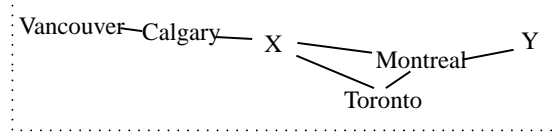
such as walk, run and jump. In the present case, the behaviour jump could be a very complex description of the mechanics of a horse jumping. Now when we want refer to a particular horse, we only need to insert the values $C = white, A = 12\ years, S = 80\ km/hour$ in our Horses class. Our horse would also come with the behaviours of the class and thus make the horse more than a list of attributes. In this setting, we don't have to redefine each new horse we are referring to, we only need to set values to the characteristics contained in the class. In programming, a horse would be said to be an instance or an objet of the Horses class.

In the language of forms and transformations, we can include the powerful property of object-oriented programming by naming a form or transformation with a label followed by a list of values written under parentheses as $Label(V_1 = value, V_2 = value, \dots, V_n = value)$. We will refer to these values as the *variables*. This is related to what is done when defining a function by writing $f(x)$ to refer to an algebraic expression with a variable x . Although we will often use the notation $Label(V_1 = value, V_2 = value, \dots, V_n = value)$, we will not restrict the language to allow only this form of label. We will see below examples where there is no restriction on the form of label. This will be convenient to represent arithmetic operations such as addition and multiplication.

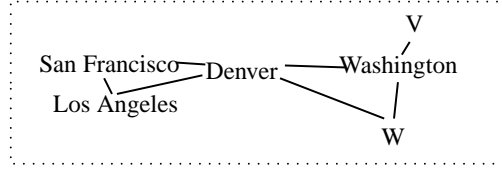
An example with one variable is $Growth(Tree) := \boxed{Tree \triangleright Seed}$. This is interpreted as turning *Tree* into a variable. Now, if we want to refer to the growth of an oak tree or a maple tree we would respectively write $Growth(Oak) :=$ and $Growth(Maple)$. An example with two variables is $Growth(Tree, Sun) := \boxed{Tree \triangleright Seed, Sun}$. Thus, if we want to refer to the growth of an oak under the midday sun, we would write $Growth(Oak, Middaysun)$. It is important to note that these names such as $Growth(Oak, Middaysun)$ are also considered to be forms, since we could also apply transformations to these names. This name associated to a form containing variables will be called a *function form*.

For a more complex example, let's define a form for the Canadian postal network. Note that we will use dotted lines to surround the collection of forms to be named.

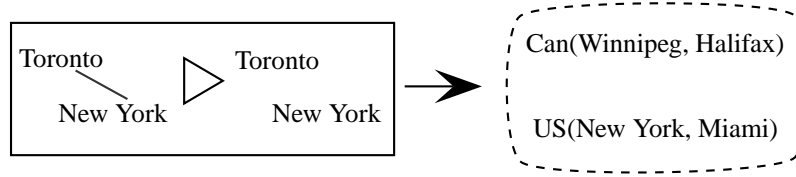
Can(X,Y):=



US(V, W):=



We can now rewrite the transformation 1 of section 4.2.8 as



A change to these networks can be done only by changing the names under the parentheses. For example, we could change New York to Boston just by writing $US(Boston, Miami)$ instead of $US(NewYork)$.

4.3.3 Abstraction

We have constructed transformations that resemble object programming classes and mathematical functions. We can start with a form and turn any part of that form into a variable. Such a process will be called form *abstraction*. When abstracting a symbol, all occurrences of that symbol in the form become a variable. For example, the initial form (A, B, B, C, A, A) can be abstracted by writing $f(A) = (A, B, B, C, A, A)$. This means that A is now considered to be a variable and that A can be replaced by any form. Example of instances of $f(A)$ are $f(E) = (E, B, B, C, E, E)$ and $f(\boxed{M \triangleright N}) := (\boxed{M \triangleright N}, B, B, C, \boxed{M \triangleright N}, \boxed{M \triangleright N})$. It is important to note that when giving instances, we will not write the symbol $:=$ but will write only $=$.

We can also use colors if there is more than one element which has the same name and we want to abstract only one of these. For example, $g(\textcolor{blue}{A}) := (A, B, B, C, \textcolor{blue}{A}, A)$ will be evaluated as $g(E) = (A, B, B, C, E, A)$. If we want to abstract to have two variables we write $h(\textcolor{blue}{A}, \textcolor{green}{B}) := (A, B, \textcolor{green}{B}, C, A, \textcolor{blue}{A})$ and evaluate as $h(E, F) = (A, B, F, C, A, E)$. If the variables have the same name, this is done by using different colors so that $k(\textcolor{blue}{A}, \textcolor{green}{A}) := (\textcolor{blue}{A}, B, B, C, A, \textcolor{green}{A})$ will be evaluated as $k(E, F) = (E, B, F, C, A, F)$.

If we want to abstract a group of elements we can use a single color for each element. For example, $j(\textcolor{green}{A}, \textcolor{green}{B}, \textcolor{green}{B}) := (\textcolor{green}{A}, \textcolor{green}{B}, \textcolor{green}{B}, C, A, A)$ will be evaluated as $j(E) = (E, C, A, A)$ or $j(E, F) = (E, F, C, A, A)$.

Abstraction can also be made on any symbol. Here are three examples.

$$\textit{Arrow}(\rightarrow) := \boxed{\textit{Tree} \triangleright \textit{Seed}}^{\textcolor{blue}{3S}} \rightarrow (\textit{Seed}, \textit{Seed}, \textit{Seed}, \textit{Soil}).$$

This can be useful to stop the application of a transformation by removing the arrow by writing $\textit{Arrow}(\quad)$.

$$\textit{Superscript}(\textcolor{green}{3S}) := \boxed{\textit{Tree} \triangleright \textit{Seed}}^{\textcolor{green}{3S}} \rightarrow (\textit{Seed}, \textit{Seed}, \textit{Seed}, \textit{Soil}).$$

This can be useful to change the number of applications and series to parallel transformations by writing $\textit{Superscript}(2P)$.

$$\textit{LettersAI}(\textcolor{green}{a}, \textcolor{blue}{i}) := \textit{Can}(\textcolor{green}{Winnipeg}, \textcolor{blue}{Hali fax})$$

This means that $\textit{LettersAI}(A, I)$ gives

$$[\textit{CAn}(\textit{WInnIpeg}, \textit{HAlifax})$$

However, in this case there is no meaning for this abstraction, we are allowing this because we want to have a very flexible language. Restrictions can always be applied when needed depending on the context. We now have a lot of flexibility with our notation and can modify almost all symbols in a transformation. The goal behind such a flexible notation is to be able to bridge the gaps between different domains of science by making sure that all can be represented with the use of transformations and useful naming process.

Until now, the variables which were abstracted were written under parentheses like is usually done with functions, but we do not need to follow this

way of naming. We can abstract forms which are similar to the operation of addition and multiplication, respectively $a + b$ and $a \cdot b$.

Integers can be represented by an amount of black dots. Thus, $1 := \bullet$, $2 := \bullet, \bullet$, $3 := \bullet, \bullet, \bullet$ and so on. We can represent $1 + 2$ as a system of transformations by writing

$$1 + 2 := \boxed{\bullet \triangleright} \rightarrow \{\bullet, \bullet\}.$$

We can abstract this to define the addition operation by writing

$$\bullet + \bullet, \bullet := \boxed{\bullet \triangleright} \rightarrow \{\bullet, \bullet\}.$$

This way, if we want to find the result of $2 + 3$, we only have that

$$2 + 3 := \boxed{2 \triangleright} \rightarrow \{3\}.$$

By replacing 2 and 3 with dots on the right hand-side, we find

$$2 + 3 := \boxed{\bullet, \bullet \triangleright} \rightarrow \{\bullet, \bullet, \bullet\},$$

which will reduce to $\{\bullet, \bullet, \bullet, \bullet, \bullet\}$ or the name 5.

Similarly, we can define multiplication by

$$a \times b := \boxed{a \triangleright \bullet}^{\#S} \rightarrow \{b\}.$$

Recall that the superscript $\#S$ indicates that the transformation disappears after it is applied until each element of the initial form has been replaced. Note that only the elements of the initial forms are replaced not the elements that are introduced during the reduction. In this case, this is not viewed as an abstraction, but only a definition. For $2 \times 3 = \boxed{2 \triangleright \bullet}^{\#S} \rightarrow \{3\}$, we find the following reduction.

$$\begin{aligned} & \boxed{2 \triangleright \bullet}^{\#S} \rightarrow \{3\} \\ & = \\ & \boxed{\bullet, \bullet \triangleright \bullet}^{\#S} \rightarrow \{\bullet, \bullet, \bullet\} \\ & \Downarrow \\ & \{\bullet, \bullet, \bullet, \bullet, \bullet\} \end{aligned}$$

4.4 Reduction view

When we are more interested in the result of each step of a reduction, it is possible to use a convenient notation. When a form X is reduced to a form Y by a transformation, we can write the name of the transformation above the reduction double arrow to indicate the transformation which was used in that reduction.

Definition. Let $T := \boxed{X \triangleright Y}$, then the following three notations are equivalent.

$$\boxed{X \triangleright Y} \rightarrow Z \Rightarrow W,$$

$$T \rightarrow Z \Rightarrow W,$$

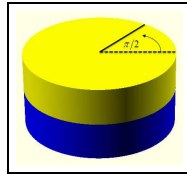
$$Z \xRightarrow{T} W.$$

We can also name steps of a reduction by using the naming symbol $:=$ written over the reduction double arrow. For example, if we have the reduction $A \Rightarrow B \Rightarrow C \Rightarrow D$, we can name the step between C and D as the transformation t by writing

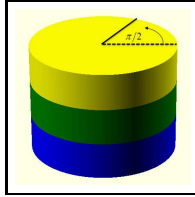
$$A \Rightarrow B \Rightarrow C \xRightarrow{t:=} D.$$

4.5 Three dimensional language

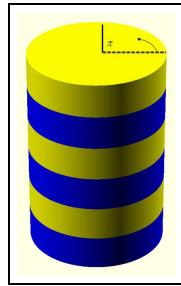
The language is not restricted to a 2 dimensional notation, but can also use a 3 dimensional notation. There are many advantages in using a 3d representation. One advantage is that instead of using subscript and superscript, we can use different shapes to represent different types of transformations. In some cases, we could quickly identify the transformation to be applied. Examples of this, from an inline notation to a 2d notation is the Penrose tensor diagram notation and Feynman diagrams. For example, a transformation which rotates an object by $\pi/2$ radians and doubles the height of an object could be named as follows.



A transformation which rotates an object by $\pi/2$ radians, but triples the height of an object could be named as follows.



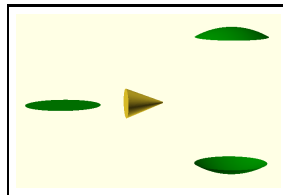
Naming a transformation with a 3D shape can be interesting if we also define a way to combine the names of similar transformations together in a geometrical way. For example, applying the tripling transformation to the geometrical shape representing the doubling transformation, we will find the following transformation.



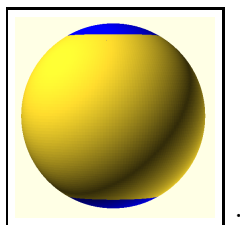
This transformation is interpreted as scaling the figure by a factor of 6 and rotating by π radians. Applied on a geometrical shape such as a tetrahedron, this is equivalent to first applying the doubling transformation and then applying the tripling transformation.

Another advantage is that we can now have a mathematical notation to study and modify geometrical objects. In mathematics, this is related to the field of topology. We will now give an example where an empty sphere with identified poles is reduced to a torus through a sequence of transformations.

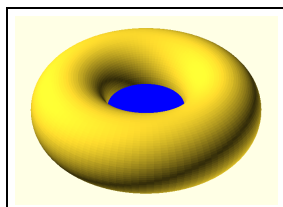
We can apply the transformation



on the empty sphere with identified blue poles given by

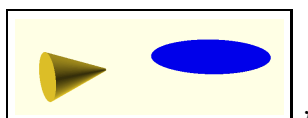


This gives the geometric object

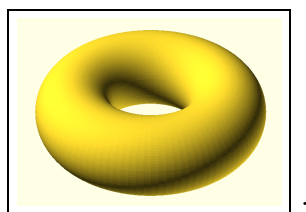


Note that since the connections need to be kept, the surface of the sphere must follow the circle in the middle of the sphere.

Then, by applying the transformation



which dissolves the blue circle, we get the following torus.



A three dimensional notation could be the new way to present scientific and mathematical ideas. It is possible that scientific papers of the future won't take the form of a written paper, but will be presented in a 3D virtual environment. The preparations and computations involved in such papers might be better accomplished in an interactive dynamic 3D virtual environment.

4.6 Diagram versus Inline Notation

Until now we have used a combination of two dimensional notations and inline notations (which is a notation which can be written as a line of text). The advantage of this is that it is sometimes faster to write and can be the basis of a computing language. The disadvantage is that it is difficult to refer to geometric properties such as angles and shapes.

Eventually, all 2D or 3D diagrams should be able to be encoded inline; although the inline notation might be hard to read; that language is closer to classical mathematics and programming. The best would be to become fluent in diagram and inline notation in a way which would allow the use of the strengths of each notation.

We now introduce the symbol “!” which when placed immediately before similar symbols will indicate that we are referring to the same symbol. For example, in the set $\{x, x, x\}$ we have three occurrences of the symbol x , but in the set $\{!x, !x, x\}$ there are exactly two occurrences of the symbol x . This is a useful notation to write geometric shapes inline.

For example, the triangular graph  can be written inline as $! \bullet - \bullet - \bullet - ! \bullet$.

4.7 Equations and solutions

One of the first mathematical equations we can think of is $2 + 2 = 4$. This is an equation since we agree that the symbols $2 + 2$ mean exactly 4 or that the statement $2 + 2 = 4$ is true. One key thing about equations is that we can also put a variable x in an equation and then ask to find the value of x which will make the equation true. In this case, the value of x would be known as a solution of the equation. For example, if we want to solve the equation $2 + 2 = x$, we can say that 4 is a solution of this equation. A solution of the equation $2 + x = 10$ would be 8. There is not always a unique solution, for example $x^2 = 4$ has 2 and -2 as solutions.

Equations are not only restricted to mathematics, but are in fact quite common. Day to day questions such as ‘What is the time?’ and ‘How old are you?’ can also be considered to be equations. For example, we can view the question ‘What is the time?’ as being asked to solve the equation Present time = x and ‘How old are you?’ as solving for x the equation $x = (\text{Today's date}) - (\text{Date of your birth})$.

Equations

Equations and solutions to equations are an essential part of a language. Finding a remedy or discovering a more precise biological process can be viewed as finding the solution of equations involving transformations. Because systems of transformations can also be reduced, our equations will take a different form. For now we will only define three types of equations, one involving the naming symbols and the other involving the reduction symbol. The equations $a := b$, $a =: b$, $a ::= b$ and $a = b$ will mean that the label a can be used instead of b in any circumstance. Note that after naming a transformation by using $a := b$, we can stop using the colon symbol and write $a = b$. The second type of equation is $x \Rightarrow y$ which can be read as ‘the system of transformations x reduces to y ’. If the system y also reduces to x we will write $x \Leftrightarrow y$. An example of this is when the two systems are the same but with different labels and names. Note that $x \Leftrightarrow x$ is always true.

If both x and y reduces to the same form c we will write $x \Rightarrow c \Leftarrow y$. This is also considered to be an equation. For example, the following two systems S_1 and S_2 reduce to the same form $(abab)$.

$$S_1 := \boxed{ab \triangleright c}^{2S} \rightarrow (cc)$$

$$S_2 := \boxed{abab \triangleright b} \rightarrow (b)$$

Thus we can write the equation

$$S_1 \Rightarrow (abab) \Leftarrow S_2.$$

When we have such an equation, we will say that the system S_1 and S_2 are *reduction equivalent* and write this as $S_1 \asymp S_2$. This is our third type of equation.

Usually, equations will be expressions involving variables, forms, transformations, systems, naming symbols, the double arrow reduction symbols and the symbol ‘ \asymp ’. Since reductions do not always terminate, it would eventually be interesting to define new types of relations to help compare different systems.

Solutions

We now present two short examples of solutions of equations. We will look at solutions to equations in more detail in section 10. Now that we have

equations (or reductions) involving forms, we can include variables in an equation and ask which forms are solutions to this equation. For example, with X as a variable, we can wonder which form will satisfy the equation

$$S_1 := \boxed{ab \triangleright X}^{2S} \rightarrow (cc) \Rightarrow (abab).$$

The solution to this equation is $X = c$. In the case of the equation

$$S_1 := \boxed{ab \triangleright X}^{2S} \rightarrow (efef) \Rightarrow (abab),$$

the solution would be $X = ef$.

4.8 Finer and coarser models

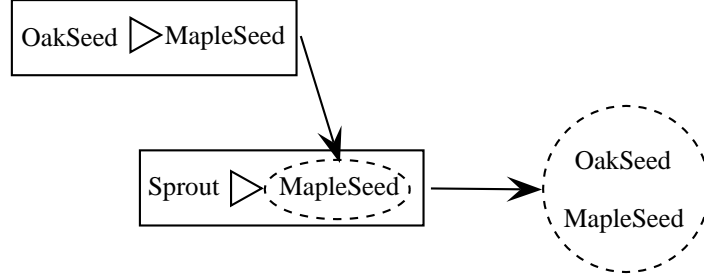
We now give an example of finer and coarser model. Take the transformation $T(C, A) := \boxed{C \triangleright A}$. Applied to the form (A) we get (C) . Applying the transformation $T(B, A) := \boxed{B \triangleright A}$ followed by $T(C, B) := \boxed{C \triangleright B}$ to (A) will reduce also to (C) . This means that $T(C, A) \Rightarrow (A)$ is reduction equivalent to $T(C, B) \rightarrow T(B, A) \rightarrow (A)$. Since the two systems are initially both applied on the same initial form, we will also say that $T(C, B) \rightarrow T(B, A)$ is a *finer* model than $T(C, A)$ or that $T(C, A)$ is a *coarser* model than $T(C, B) \rightarrow T(B, A)$.

This can be seen as having refined the process of changing A into B by writing the subprocess involved in this coarser change.

4.9 Higher Order

We have seen that each initial form is affected by a collection of transformations and how reductions are accomplished when there is one initial form. We now extend our language to include multiple initial forms in a system such that transformations which are applied to an initial form can themselves be affected by other transformations, thus making the language a higher order language. Moreover, a transformation will also be able to affect itself, thus allowing recurrence relations and transformations.

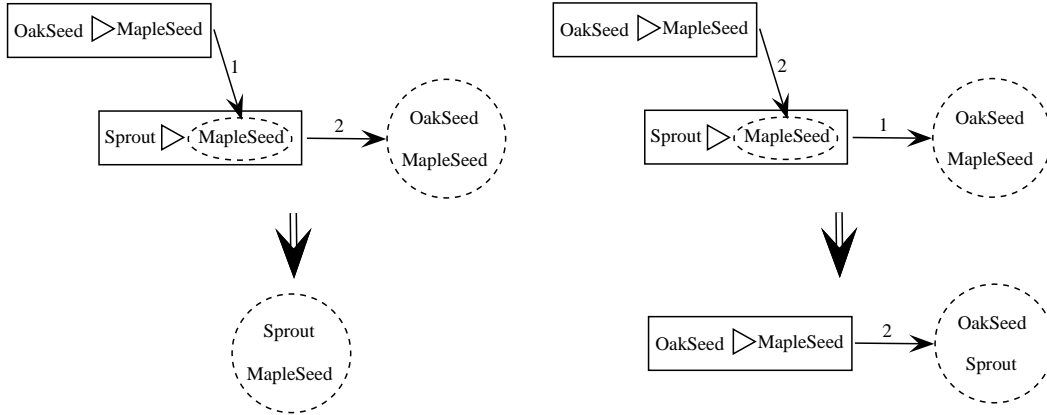
Let's look at a first example.



We notice that depending on which transformation is applied first, we will have a different type of seed sprouting. We thus have to introduce a new notation to indicate the order in which the arrows are applied.

Numbers over the arrows indicate the order in which each arrow is applied. The arrow with the smallest number is applied first. It is understood that there is no number for series of transformations since there is only one initial form and the closest transformation to the initial form is applied first. Similarly to parallel transformations, if there are no numbers on the arrows or the same number on multiple arrows, then the choice of the applied transformation is arbitrary.

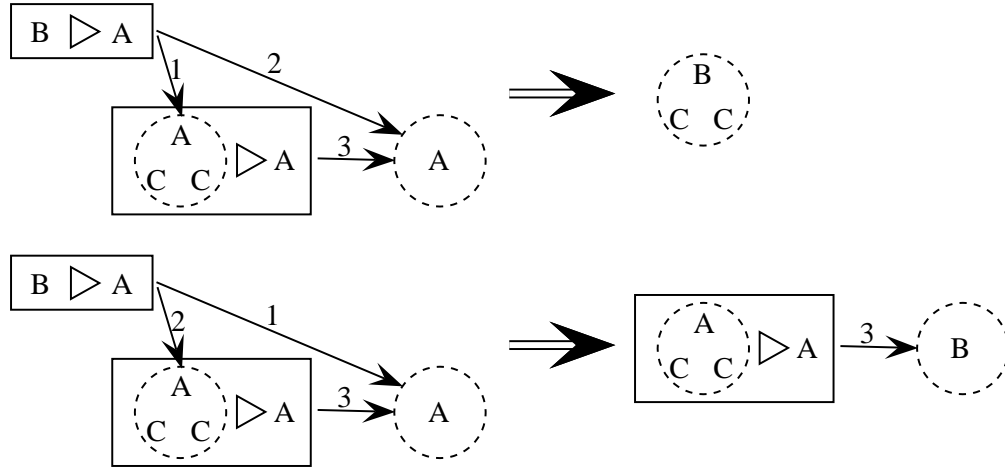
Here are the reductions for different numberings. Note that if both arrows are not numbered or numbered with the same number, the reduction will be an arbitrary reduction between these two.



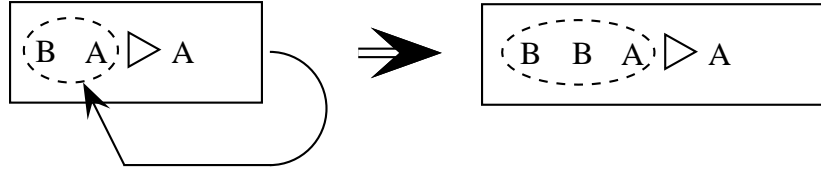
Having access to higher order transformations can be interesting to model complex systems and also for testing of systems. When building a first order

model, collections of transformations applied to the whole model can help refine the model.

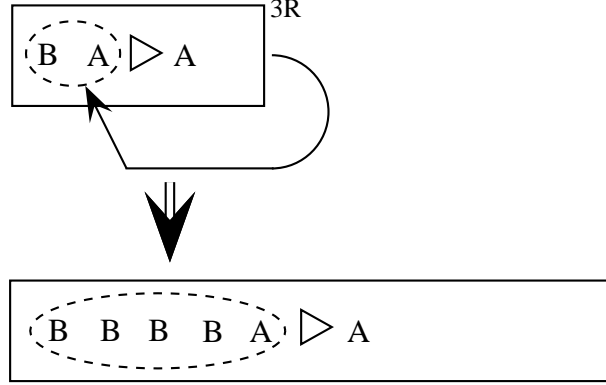
We can also have a transformation point to two different initial forms. Note that when a transformation points to two forms, it will disappear after it was applied unless there is a number at the superscript indicating the presence of multiple copies of the transformation.



We now show an example of a transformation affecting itself.



In the following example, we have a transformation which is applied 3 times to itself. To denote this, we will use a superscript with a number followed by the letter *R* indicating a recurrence.



5 Probabilities

When we have two transformations applied in parallel to an initial set, we are naturally faced with the question of which transformation should be applied. We can take advantage of this by introducing the concept of probability in our language. Take the transformation

$$[\boxed{heads \triangleright coin}, \boxed{tails \triangleright coin}] \rightarrow \{coin\}$$

As we already mentioned, we will interpret the transformation as saying that there is a 1 chance out of 2 that the initial set containing a single coin is reduced to heads and a 1 chance out of 2 (or 0.5 chance) that the initial set is reduced to tails. The chances of being applied can also be indicated by using numbers on the subscript of each transformation as follows.

$$[\boxed{heads \triangleright coin}_{0.5}, \boxed{tails \triangleright coin}_{0.5}] \rightarrow \{coin\}$$

For more occurrences of each transformation, we have to differentiate between series and parallel transformations. For example,

$$[\boxed{heads \triangleright coin}^{2P}, \boxed{tails \triangleright coin}] \rightarrow \{coin\}$$

is understood as each of the three transformation having 1/3 chance of being applied.

For series of transformations, the example

$$[\boxed{heads \triangleright coin}^{2S}, \boxed{tails \triangleright coin}^{2S}] \rightarrow \{coin, coin\}$$

is understood as a 1/2 chance for each transformation to be applied first, but before the same transformation is applied a second time, all other transformations need to be applied once if they can be applied. This system will automatically result

$$[\boxed{heads \triangleright coin}^{1S}, \boxed{tails \triangleright coin}^{1S}] \rightarrow \{heads, tails\}$$

The following system

$$[\boxed{heads \triangleright coin}^{2S}, \boxed{tails \triangleright coin}^{2S}] \rightarrow \{coin, coin\}$$

has 1/2 chance of reducing to

$$[\boxed{tails \triangleright coin}^{2S}] \rightarrow \{heads, tails, heads\}$$

and 1/2 chance of reducing to

$$[\boxed{heads \triangleright coin}^{2S}] \rightarrow \{heads, tails, tails\}.$$

Another example is

$$[\boxed{1 \triangleright dice}_{\frac{1}{6}}, \boxed{2 \triangleright dice}_{\frac{1}{6}}, \boxed{3 \triangleright dice}_{\frac{1}{6}}, \boxed{4 \triangleright dice}_{\frac{1}{6}}, \boxed{5 \triangleright dice}_{\frac{1}{6}}, \boxed{6 \triangleright dice}_{\frac{1}{6}}] \rightarrow \{dice\}.$$

We will interpret the transformation as saying that there is a $\frac{1}{6}$ chance of getting any of numbers 1 to 6 on one roll.

The transformation

$$[\boxed{heads \triangleright coin}_{\frac{1}{3}}, \boxed{tails \triangleright coin}_{\frac{1}{3}}, \boxed{tails \triangleright coin}_{\frac{1}{3}}] \rightarrow \{coin\}$$

is interpreted as 1/3 chance of getting heads and 1/3 chance of getting tails for each tails, for a total of 2/3 chance of getting tails. So in general, if n is the number of parallel transformations, then there is 1/ n chance of that transformation to be applied. Note that in this case also, we do not need to use subscripts since no subscript on parallel transformations indicates similar chances of being applied.

We now add a new subscript notation, which will give different probabilities to each of the parallel transformations. The transformation

$$[\boxed{heads \triangleright coin}_{0.1}, \boxed{tails \triangleright coin}_{0.9}] \rightarrow \{coin\}$$

is interpreted as saying that there is a 10% chance that the initial set containing a single coin is reduced to heads and a 90% chance that the initial set is reduced to tails. Note that if there is no subscript on the parallel transformations, then we consider they have the same chances of being applied and as mentioned above; have a probability of $1/n$.

The sum of the probabilities does not necessarily need to be 1 as in our example $0.1 + .90$. If the sum is S and is smaller than 1, we will consider that the remainder $1 - S$ is the probability that no transformation is applied. For example,

$$[\boxed{heads \triangleright coin}_{0.25}, \boxed{tails \triangleright coin}_{0.15}] \rightarrow \{coin\}$$

means that there is a $0.6 = 1 - (0.25 + 0.15)$ chance that a flipped coin does not reach the ground or that the coin lands on its side, thus not giving a heads or tails value.

Each subscript does not need to be smaller than 1. If we have

$$[\boxed{heads \triangleright coin}_1, \boxed{tails \triangleright coin}_1] \rightarrow \{coin, coin\},$$

this means that the two transformations will be applied simultaneously and will give $\{heads, tails\}$. When applying transformations simultaneously, there there is a possibility of conflict. For example, if we have

$$[\boxed{heads \triangleright coin}_1, \boxed{tails \triangleright coin}_1] \rightarrow \{coin\},$$

we should apply both transformations simultaneously, but there is only one coin. This will be reduced to $\{heads \wedge tails\}$, which is read ‘both heads and tails’. Importantly, this can only happen if we cannot avoid the conflict, meaning that $[\boxed{heads \triangleright coin}_1, \boxed{tails \triangleright coin}_1] \rightarrow \{coin, coin\}$, will not reduce to $\{heads \wedge tails, coin\}$ because there are enough coins in the initial set.

If we have

$$[\boxed{heads \triangleright coin}_{1.1}, \boxed{tails \triangleright coin}_{1.9}] \rightarrow \{coin, coin, coin\},$$

we will interpret this as applying simultaneously the two transformations and then applying once one of the two transformations with respective probabilities of 0.1 and 0.9. Thus, this expression will reduce after the first step to

$$[\boxed{heads \triangleright coin}_{1.1}, \boxed{tails \triangleright coin}_{1.9}] \rightarrow \{heads, tails, coin\},$$

which will reduce after the second step to $\{heads, tails, heads\}$ with a 0.1 chance and $\{heads, tails, tails\}$ with a 0.9 chance.

We can also use the probability notation in a series of transformations.

$$\boxed{heads \triangleright coin}_{0.85} \rightarrow \boxed{tails \triangleright coin}_{0.2} \rightarrow \{coin\}$$

means that there is 0.2 chance of the coin becoming tails after the first step and 0.8 chance of staying a coin. At the second step, there is 0.85 chance of the coin becoming tails after the first step and 0.15 chance of staying a coin.

6 Summary of the notations

We now give a summary of the notations which were defined above.

Basic notation

- Any symbol or collection of symbols is called a form. There are no restrictions on what a form can be. Example of forms are strings of letters, diagrams and three dimensional objects.
- $\boxed{Y \triangleright X}$ is a transformation and represents the potential to replace X by Y .
- The right-hand side of the symbol \triangleright is called the cause of the transformation and the left-hand side is called effect of the transformation.
- The initial form is the form to which the transformations are applied.
- $\boxed{Y \triangleright X} \rightarrow (Z)$ is said to be a system.
- A system can be reduced when the right-hand side of the \triangleright symbol corresponds to an element in the initial form. Reduction is indicated with the symbol \Rightarrow . For example, $\boxed{Y \triangleright X} \rightarrow (A, X, B) \Rightarrow (A, Y, B)$.

Naming

- Names are given by using the notation $:=$, $=:$ and $:=:$ where the position of the colon indicates the side where the name appears.
- Identifier is usually only a word to designate a form. For example, XYtransformation $:= \boxed{Y \triangleright X}$.

- Function forms are composed of a label and variables where we can substitute the variable by forms. For example $\text{transform}(X,Y) := \boxed{Y \triangleright X}$ is evaluated at a and b by writing $\text{transform}(a,b) := \boxed{a \triangleright b}$.
- Abstraction of a form is done by taking a form and changing some elements into variables. For example, the ordered set $\{a, b, c\}$ can be abstracted in b by writing $\text{Abstraction}(X) := \{a, X, c\}$.

Series and parallel transformations

- Transformations applied in series are written as $\boxed{Y \triangleright X} \rightarrow \boxed{W \triangleright V} \rightarrow (Z)$ where the closest to the initial form is applied first.
- When n copies of a transformation is applied in series, we can denote this by the superscript nS as in $\boxed{Y \triangleright X}^{nS} \rightarrow (Z)$. We can have an infinite number of copies by writing ∞ instead of n .
- Transformations applied in parallel are written under square brackets as $\boxed{\boxed{Y \triangleright X}, \boxed{W \triangleright V}} \rightarrow (Z)$.
- When n copies of a transformation are applied in parallel, we can denote this by the superscript nP as in $\boxed{\boxed{Y \triangleright X}}^{nP} \rightarrow (Z)$.
- For series and parallel transformations, we can have an infinite number of copies of transformations by writing ∞ instead of n .
- For series and parallel transformations, the \sharp symbol on the superscript indicates that the transformations replace all they can until they cannot be applied, then the transformations disappear.

Distance and order

- If the distances or order of the elements of the initial form are important, we use ellipses with dashed lines in diagram view; and parentheses $()$ in the inline view. When the distances and order are not important, we use ellipses with full lines in diagram view and braces $\{\}$ in the inline view.
- A transformation with disjoint elements on the cause of a transformation does not require that the order or the distance between them to

be the same as in the initial form to be applied. For example, we have that $\boxed{X, Y \triangleright C, A} \rightarrow (A, B, C) \Rightarrow (Y, B, X)$

- If we write the subscript *rigid* on \triangleright , then order and distance need to be respected. For example, $\boxed{X, Y \triangleright_{\text{rigid}} B, C} \rightarrow (A, B, C) \Rightarrow (A, X, Y)$.

Colors

- Black forms at the same position in the cause and effect of a transformation indicates that the form in the cause will replace the one from the effect.
- Forms of the same color in the cause and effect of a transformation indicates that the form in the cause will replace the one from the effect.
- The forms in red in the cause of a transformation indicates that the transformation cannot be applied if the form in red appears in the initial form. Red is reserved for this interpretation.

Connections

- Connections between nodes or vertices are usually denoted by lines.
- If form A is replaced by form B, all the connections that node A has are transferred to B and B keeps its previous connections.

7 Mathematical models in medicine

We now present the main application of our language which is mathematical medicine. To enhance our motivation, have objectives and have an idea of the important points regarding the construction for an efficient mathematical language of medicine, we can keep in mind the following list of milestones to be attained.

Milestones

- A large open database incorporating all known biological processes (this is mostly a translation phase, where all papers are translated into a common mathematical framework).

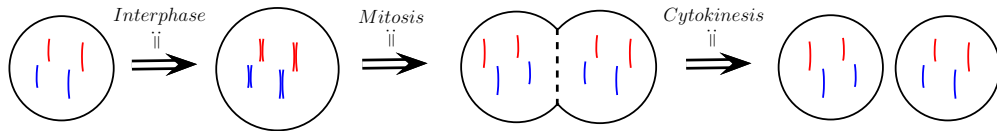
- Computing treatments with positive results.
- Discovering new processes or refining a process.
- Having a global model which makes animal testing irrelevant.
- Having individually tailored medicine.
- Widely accessible medicinal compounds which can be printed, sent or created in local laboratories.
- Curing, understanding or controlling all known diseases and conditions.
- Prediction of potential future diseases and study of theoretical biological systems in different types of environments.

We will present in the following sections different biological models ranging from DNA replication up to a beating heart.

7.1 Cell-division cycle

The division cycle of a cell with a membrane-bound nucleus can be divided into three periods: interphase, mitosis and cytokinesis. In the interphase, the DNA is duplicated, during the mitosis, the duplicated DNA strands are separated and at the end of cytokinesis, we observe two daughters cells.

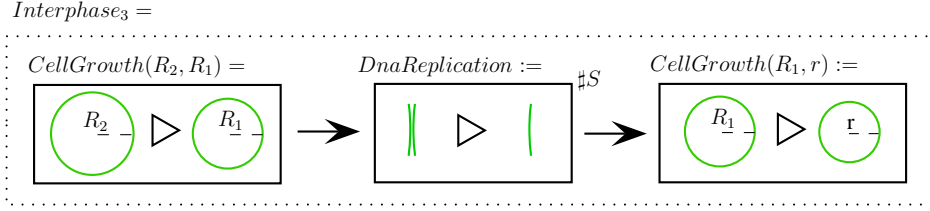
At the coarsest level, cell division can be modelled as follows. Note that in the following reduction, each transformation is named by using the “:=” symbol.



7.2 Interphase

The transformation *Interphase* can be further refined by defining the following model named *Interphase3*. Both transformations applied on an initial cell will reduce to the same form, but since *Interphase3* is composed of three

transformations instead of one, we write the number 3 next to the name interphase. As a general rule, the higher the number, the more refined the model, but this might not always be the case.

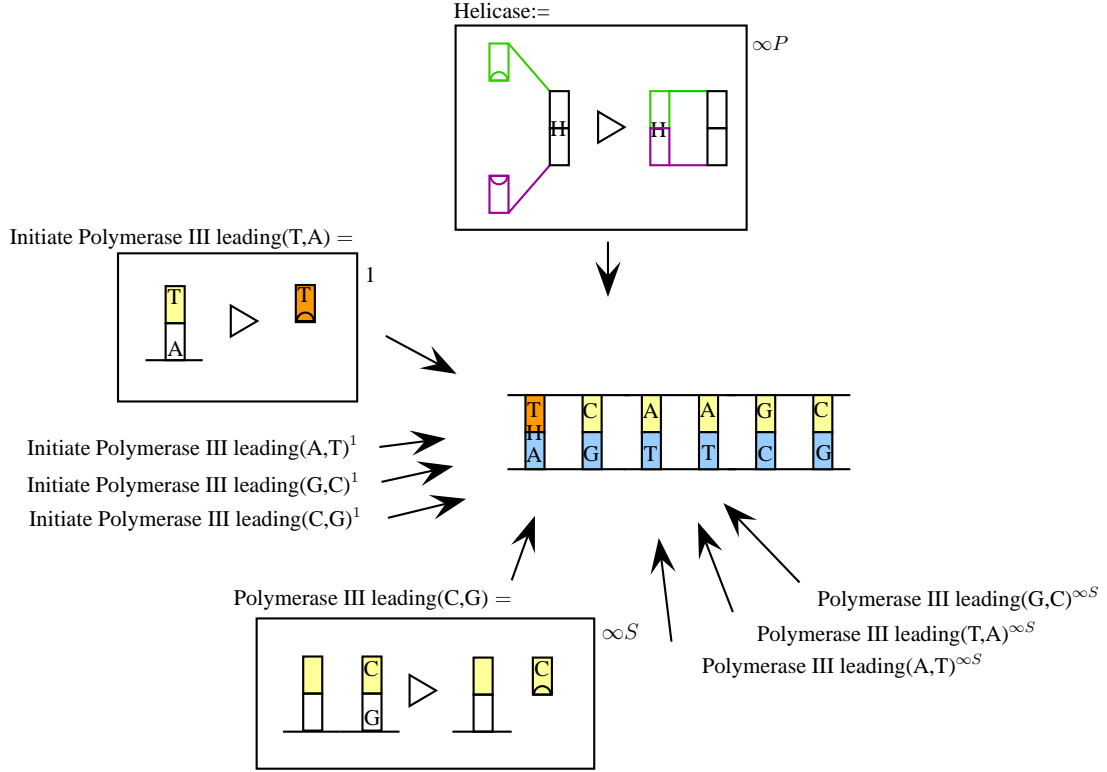


$CellGrowth(r, R_1)$ is usually called the G_1 -phase, $CellGrowth(R_1, R_2)$ is called the G_2 -phase and $DnaReplication$ is called S -phase.

7.3 DNA replication

We will now give a system that replicates DNA. The collection of transformations used in the process can be considered to be a refined model of the transformation $DnaReplication$.

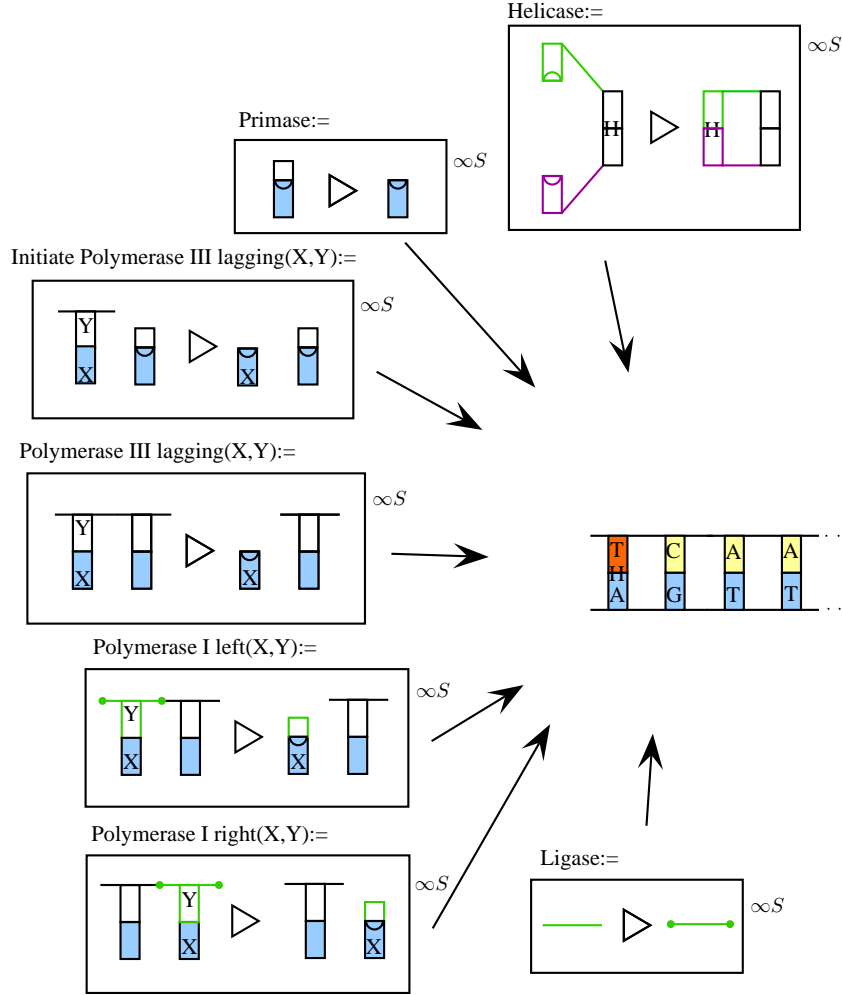
7.3.1 Diagram for the replication of the right leading strand



To avoid rewriting each type of *Initiate Polymerase III leading*, we used abstraction of the form *Initiate Polymerase III leading*(A, T). Since transformations preserve connections; the bonds between nucleotides are preserved. The letter written on the nucleotides is considered to be connected to the nucleotide than contains it, thus when nothing is written on the nucleotide, it is assumed that the letter is still on the nucleotide after the application of a transformation.

The system modeling the replication of the right lagging strand is now given. Instead of writing all combinations of pairs of nucleotides, we introduce the notation *For all* $(X, Y) \in \{(A, T), (T, A), (G, C), (C, G)\}$ which we write at the bottom of the diagram. This notation means that each transformation over (X, Y) will come in four copies, that is one for each ordered pair. Note that we have given another way to write the transformation for helicase.

7.3.2 Diagram for the replication of the right lagging strand



For all $(X,Y) \in \{(A,T), (T,A), (G,C), (C,G)\}$

We will now define the same right DNA replication process but written in inline notation. We will also define the left replication inline.

DNA will be encoded in an array $\langle a_1, a_2, \dots, a_n \rangle$ and in each component a_i of the array there will be a pair of nucleotides (X, Y) . The k th component is underlined to indicate where helicase will start.

$$dna := \langle (x_1, y_1) \mid (x_2, y_2) \mid \dots \mid \underline{(x_{k-1}, \overleftarrow{y}_{k-1})} \mid \underline{(\overrightarrow{x}_k, y_k)} \mid (x_{k+1}, y_{k+1}) \mid \dots \mid (x_n, y_n) \rangle$$

We now define helicase and polymerase. Note that in our language, if the name starts with a lowercase, it will refer to the inline notation and if the name starts by an uppercase, it will refer to the diagram notation.

7.3.3 Inline replication of the right leading strand

Note that in the following transformations, we use an asterisk ‘*’ to indicate an empty component of a pair.

$$\text{helicase right} := \boxed{(x_i, *)(*, y_i) \mid \underline{(x_{i+1}, y_{i+1})} \triangleright \underline{(x_i, y_i)} \mid (x_{i+1}, y_{i+1})}$$

$$\text{helicase final right} := \boxed{(x_i, *)(*, y_i) \rangle \triangleright \underline{(x_i, y_i) \rangle}}$$

$$\text{initiate polymerase leading right}(X, Y) := \boxed{(X, Y) \triangleright (\vec{X}, *)}$$

$$\text{polymerase leading right}(X, Y) := \boxed{(K, L)(M, N) \mid (X, Y) \triangleright (K, *) (M, N) \mid (X, *)}$$

Where K, L, M and N can be any nucleotide or the symbol *, but where * means that it cannot be the asterisk symbol. We have a transformation *for all* $(X, Y) \in \{(A, T), (T, A), (G, C), (C, G)\}$, *for all* $(V, W) \in \{(A, T), (T, A), (G, C), (C, G)\}$.

7.3.4 Inline replication of the left leading strand

$$\text{helicase left} := \boxed{\underline{(x_{j-1}, y_{j-1})} \mid (x_j, *)(*, y_j) \triangleright (x_{j-1}, y_{j-1}) \mid \underline{(x_j, y_j)}}$$

$$\text{helicase final left} := \boxed{\langle (x_j, *)(*, y_j) \triangleright \langle (x_j, y_j) \rangle}$$

$$\text{initiate polymerase leading left}(X, Y) := \boxed{(X, Y) \triangleright (*, \overleftarrow{Y})}$$

$$\text{polymerase leading left}(X,Y) := \boxed{(X,Y) \mid (K,L)(M,N) \triangleright (*,Y) \mid (K,L)(\textcolor{red}{*},N)}$$

After applying all the previous transformations on the leading strand, we find the following structure which is the replicated leading strand.

$$\begin{aligned} \text{repLeading} := & \langle (x_1,*)(x_1,y_1) \mid (x_2,*)(x_2,y_2) \mid \dots \mid (x_{k-1},*)(x_{k-1},y_{k-1}), \\ & (x_k,y_k)(*,y_k) \mid (x_{k+1},y_{k+1})(*,y_{k+1}) \mid \dots \mid (x_n,y_n)(*,y_n) \rangle \end{aligned}$$

7.3.5 Inline replication of the right lagging strand

$$\text{primaseR} := \boxed{(\square, Y) \triangleright (*, Y)}$$

$$\text{initiate poly III lagging right}(X,Y) := \boxed{(X,Y) \mid (\square, B) \triangleright (*, Y) \mid (\square, B)}$$

$$\text{polymerase III lagging right}(X,Y) := \boxed{(X,Y) \mid (A,B) \triangleright (*, Y) \mid (A,B)}$$

$$\text{polymerase Ia lagging right}(X,Y) := \boxed{(\tilde{X}, Y) \mid (A,B)(\textcolor{red}{\square}, E) \triangleright (\square, Y) \mid (A,B)(\textcolor{red}{\square}, E)}$$

$$\text{polymerase Ib lagging right}(X,Y) := \boxed{(A,B)(D,E) \mid (\tilde{X}, Y) \triangleright (\textcolor{red}{\square}, B)(D,E) \mid (\square, Y)}$$

$$\text{ligase} := \boxed{X \triangleright \tilde{X}}$$

7.3.6 Inline replication of the left lagging strand

$$\text{primaseL} := \boxed{(X, \square) \triangleright (X, *)}$$

$$\text{initiate poly III lagging left}(X, Y) := \boxed{(A, \square) \mid (X, Y) \triangleright (A, \square) \mid (X, *)}$$

$$\text{polymerase III lagging left}(X, Y) := \boxed{(A, B) \mid (X, Y) \triangleright (A, B) \mid (X, *)}$$

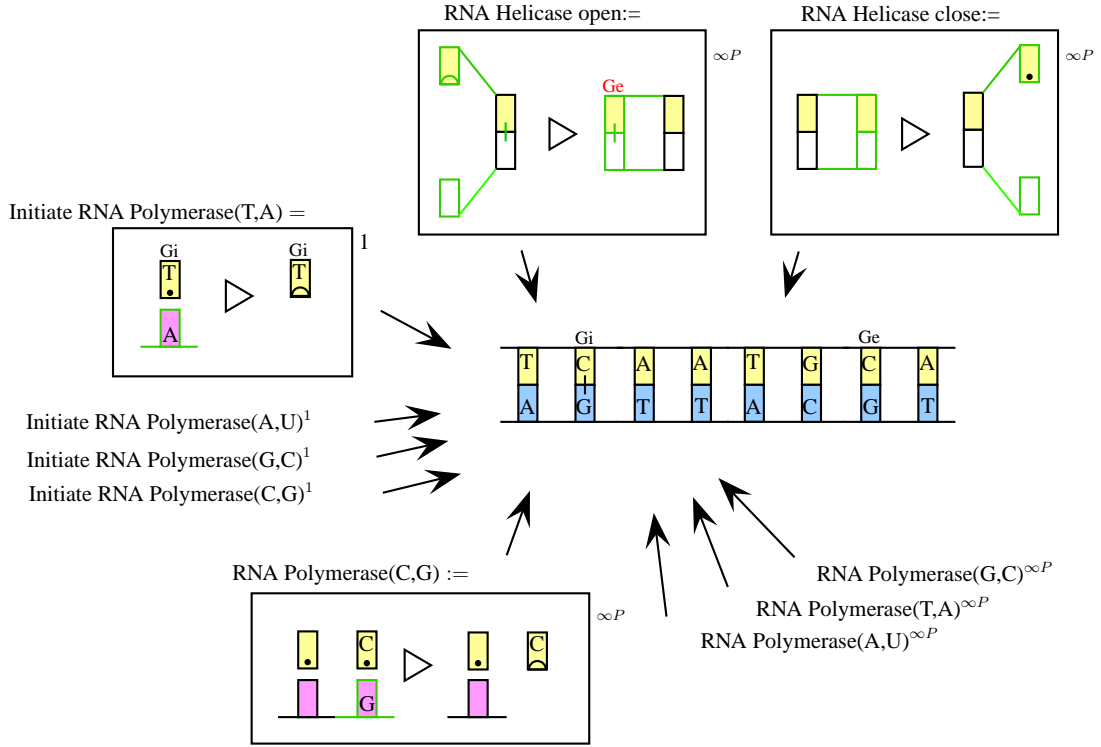
$$\text{polymerase Ia lagging left}(X, Y) := \boxed{(A, B)(D, E) \mid (X, \tilde{Y}) \triangleright (A, \square)(D, E) \mid (X, \square)}$$

$$\text{polymerase Ib lagging left}(X, Y) := \boxed{(X, \tilde{Y}) \mid (A, B)(D, E) \triangleright (Y, \square) \mid (A, B)(D, \square)}$$

$$\text{ligase} := \boxed{Y \triangleright \tilde{Y}}$$

7.4 Messenger RNA

The transcription of DNA into pre-messenger RNA is very similar to DNA replication of the right leading strand.



7.5 RNA Splicing

After the pre-messenger RNA has been coded, the introns must be removed by the spliceosome. We will simplify the model by assigning to the pre-mRNA three points: branch site B , 5' splice site $S5$ and the 3' splice site $S3$. These points are determined in practice by the spliceosome, pairs of nucleotides for the splice sites (GU for the 5'site and AG for the 3' site) and sequences of nucleotides for the branch site. The sites will be indicated with the use of superscripts written on the nucleotides. Note that the left-hand side of the pre-mRNA is the 5' and right-hand side is 3'.

$$\text{pre-mRNA} := A - \dots - D - E^{S5} - F - \dots - G - H^B - I - \dots - K - L^{S3} - M - \dots - Q$$

$$\text{fold} := \boxed{\begin{array}{ccc} F & \text{---} & E^{S5} \\ | & & | \\ G & \text{---} & H^B \end{array} \triangleright E^{S5} - F, G - H^B}$$

$$\text{spliceS5} := \boxed{D^* \quad E^{S5} \triangleright D - E^{S5}}$$

$$\text{join} := \boxed{\begin{array}{ccc} & D^* & \\ & | & \\ L^{S3} & \text{---} & M \end{array} \triangleright D^*, L^{S3} - M}$$

$$\text{spliceS3} := \boxed{L^{S3} \quad M \triangleright L^{S3} - M}$$

Thus the splicing of an intron is given by

$$\text{spliceS3} \rightarrow \text{join} \rightarrow \text{spliceS5} \rightarrow \text{fold} \rightarrow \{\text{pre-mRNA}\}$$

and gives two pieces of RNA. The piece with the loop is the intron and the other is a strand composed of two exons.

7.6 Proteins

After splicing, we have messenger RNA (mRNA). To differentiate between pre-mRNA and mRNA, we will identify the first nucleotide with a tilde. Here is an example:

$$\text{mRNA} := \tilde{A} - B - C - D - M - N - O - P - \dots - Z.$$

Initiation of the protein encoding is done with the transformation

$$\text{initiate protein} := \boxed{\begin{array}{ccc} \textit{init} & & \\ \parallel & \triangleright & \\ \hat{U} & & \tilde{U} \end{array}}$$

To synthesize a protein, the ribosome reads a sequence of triplets composed of different combinations of the nucleotides C, A, G and U. We will now give the general form of codons needed to synthesize a protein.

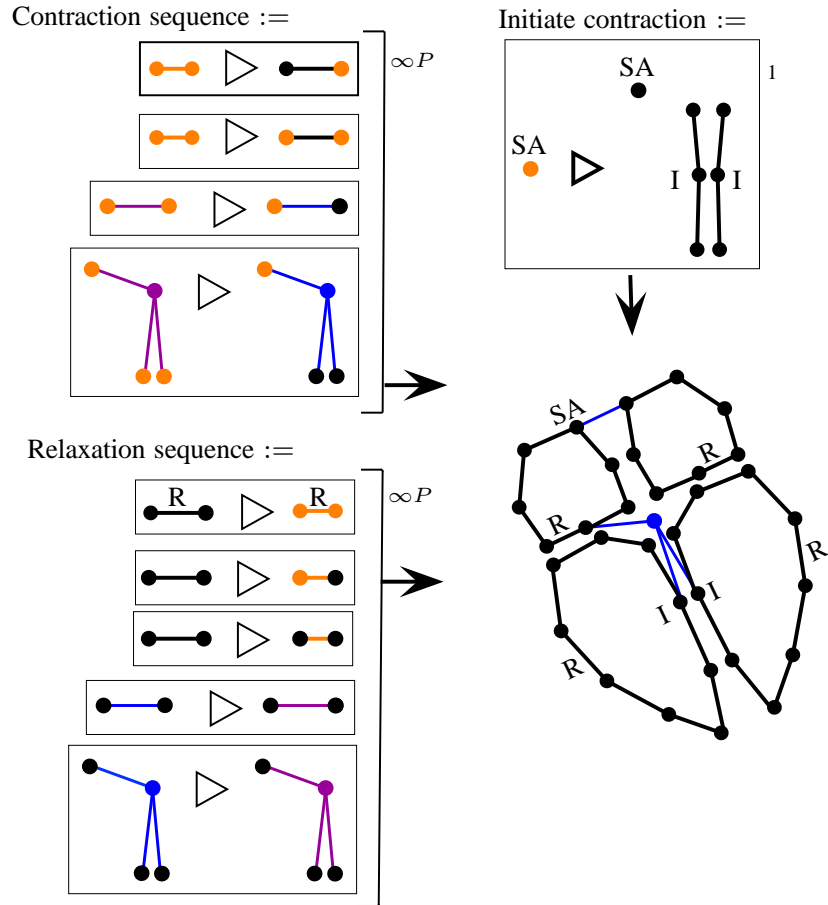
$$codon(L, UVW) := \boxed{\begin{array}{ccc} K \sim L & & K \\ \parallel & \triangleright & \parallel \\ U - V - W - \hat{X} & & \hat{U} - V - W - X \end{array}}$$

The top part with connection symbols ‘ \sim ’ is the protein being assembled and the bottom part is the RNA-messenger being decoded. The cause of the transformation is equivalent to reading the code VWX and in the effect, VWX is skipped to be in position for the next reading. Also in the effect, after reading the code VWX , the amino acid L is added on the right end of the protein next to K .

Examples of specific codons are $codon(Leucine, CCG)$, $codon(Leucine, CCC)$, $codon(Phenylalanine, UUU)$, $codon(Lycine, AAA)$ and the stop codon $codon(stop, UAG)$ which will stop the synthesis of the protein.

7.7 Heart model

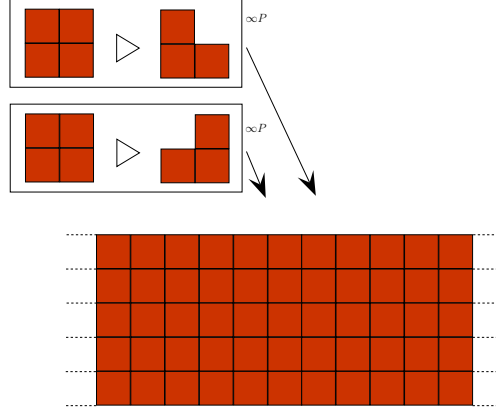
We now present a simplified model of a beating heart. The heart is composed of the right ventricle, the left ventricle, right atrium and left atrium along with the heart conduction system. The conduction system is composed of the sinuatrial node (SA), at the top in blue, we have the Bachmann’s bundle and in blue near the ventricles we have the bundle of His with left and right bundle branches. An edge connected to two nodes represents a cell (cardiac myocyte) which has the ability to contract. The cells denoted by R are markers to initiate the relaxation process. Note that in this model the distances matter in the initial forms and the transformations.



Based on the same principles, it would be possible to construct a 3D model of a beating heart. Interestingly, it would also be possible to simulate conditions like cardiac arrhythmia by putting different restrictions in the model or applying different types of transformations to the model. For example, by randomly contracting pacemaker cells of the model we could study the effects on the heart's functions.

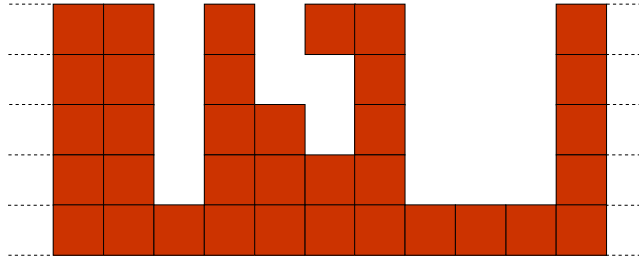
7.8 Skin Healing

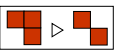
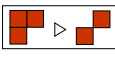
The skin's epidermis, in particular the stratum spinosum, can modeled very coarsely as seen below.



When we remove cells from our model to represent a skin injury, the two transformations will replace the cells from the bottom up. This can be interpreted as an upward migration of the keratinocytes.

The following figure is an example of a skin injury to which the two transformations can be applied.



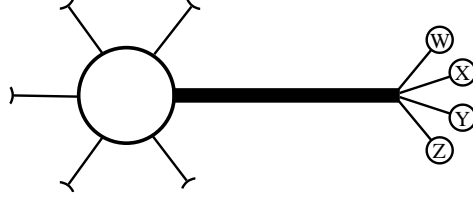
Note that the two transformations can be simplified into  and . Moreover, only one of these would be enough for our model skin healing.

7.9 Neurons

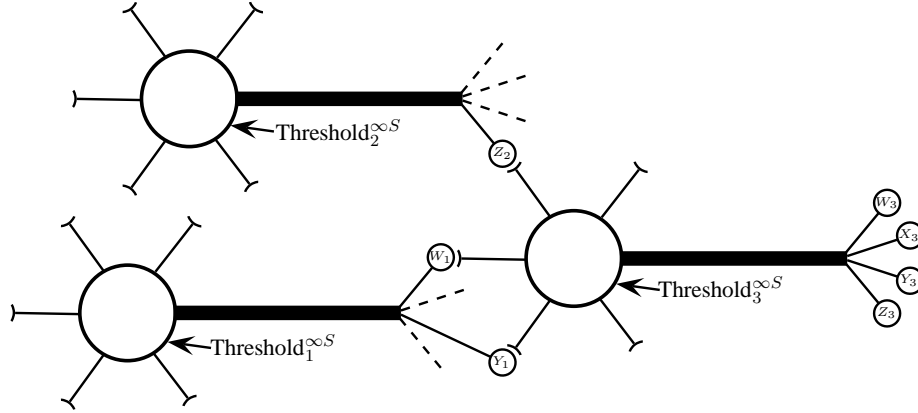
We will now give a model of a network of neurons including the transformations associated with it.

7.9.1 Diagram model

Here is a model of a single neuron.



Neurons are connected as follows. Note that we have also included the threshold transformation of each neuron.



In our simple model, we will be concerned with two actions of the neurotransmitters, excitatory (E) and inhibitory (I). Thus, The variables such as W, X, Y and Z will only take the values E or I .

Each neuron can have a different threshold. An example of the transformation which signals that the threshold has been attained is

$$Threshold := \boxed{Ready \triangleright E, E, E}.$$

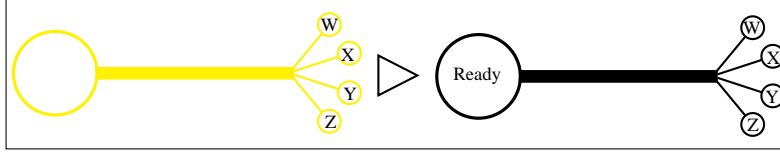
Inhibition is represented by the transformation

$$Inhibition := \boxed{\triangleright E, I}$$

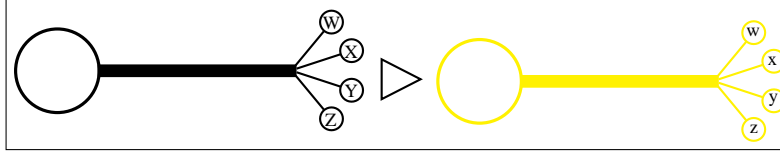
which cancels out E and I together. The initial set of these transformations is the content of the main large circle of the model neuron. There are two possibilities concerning the inhibition transformation. It can be applied in

parallel to the threshold transformation or it can be applied in series with a ‘ \sharp ’ on the superscript followed by the threshold transformation. If it is parallel, then even if there are still letters I , the threshold can be reached. If it is in series, the threshold can be reached only if there are no more letters I . Here is a list of the transformations needed.

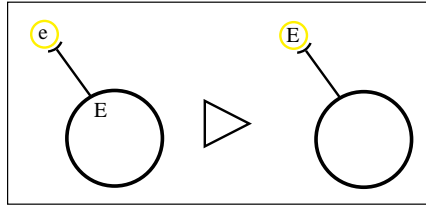
Firing :=



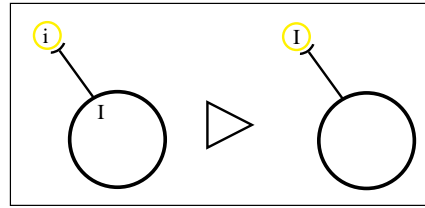
Reinitialization :=



E-contribution :=



I-contribution :=



The process of a neuron firing is as follows. It receives the E and I contributions when the input neurons are firing. Inhibition is applied in parallel or in series to the threshold transformation. When the threshold is reached, the word ‘Ready’ is written in the neuron. Then the neuron fires with the firing transformation. After this, contributions are added to other neurons and when all contributions have been added to the other neurons, we only have lowercase letters on the terminals of the axon. Then the neuron is reinitialized and will fire again when the threshold is reached.

7.9.2 Inline model

We now give an inline model of a few interconnected neurons. Each of these neurons will also have 5 input (A, B, C, D, E) and 4 output termi-

nals (W, X, Y, Z) . Each neuron will be assigned a number n written as a subscript.

$$neuron_n := [(\quad, A_n)(\quad, B_n)(\quad, C_n)(\quad, D_n)(\quad, E_n) \ll \square \gg W_n, X_n, Y_n, Z_n]$$

The letter A_n, B_n, C_n, D_n and E_n in the pairs such as (\quad, A_n) will be replaced with the terminals of other neurons when we create a network. Examples of such terminals are W_5, X_2, W_{16} and Z_{6002} . The empty left component of the pairs will be replaced by the type of connection, that is E for excitatory and I for inhibitory. Note that the order in which the pairs appear is important, that is $(\quad, A_n)(\quad, B_n)$ is not the same as $(\quad, B_n)(\quad, A_n)$. The main circle of our diagram model is written inline as $\ll \square \gg$ where a white box means that the neuron is not ready to fire, while a black box means that the threshold has been reached.

$$\text{e-contribution}_n := \boxed{(e, D_n), \ll E, \quad \triangleright \quad (E, D_n), \ll}$$

$$\text{i-contribution}_n := \boxed{(i, D_n), \ll I, \quad \triangleright \quad (I, D_n), \ll}$$

$$threshold_n := \boxed{\quad \blacksquare \quad \triangleright \quad \text{set containing letters E and I, } \square}$$

The e-contribution_n acts by transferring neurotransmitters represented by ‘ E ,’ into the center part by writing $\ll E, \square \gg$. The i-contribution_n works in the same way. An example of threshold for a neuron k is

$$threshold_k := \boxed{\quad \blacksquare \quad \triangleright \quad E, E, E, I, \square}$$

$$firing_n := \boxed{\square \gg W_n, X_n, Y_n, Z_n \quad \triangleright \quad \blacksquare \gg W_n, X_n, Y_n, Z_n}$$

$$reinitialization_n := \boxed{\ll \square \gg W_n, X_n, Y_n, Z_n \quad \triangleright \quad \ll \square \gg w_n, x_n, y_n, z_n}$$

An active neuron is the collection of the following transformations: $neuron_n$, $threshold_n$, e-contribution_n , i-contribution_n , $firing_n$ and $reinitialization_n$.

For the transformations $e\text{-contribution}_n$, $i\text{-contribution}_n$, $firing_n$ and $reinitialization_n$, only the subscript will change between neurons. So in general, to define an active neuron, we will only mention the transformations $neuron_n$ and $threshold_n$.

We now define a network of three neurons which is the same as our network of the three neurons diagram above. Note that the question mark means that we are not showing what connection it is and if we have a space as in $(\ , A_3)$, we understand that there is no connection.

$$neuron_1 := [(\ ?, \ ?)(\ ?, \ ?)(\ ?, \ ?)(\ ?, \ ?)(\ ?, \ ?) \ll \square \gg W_1, \ ?, Y_1, \ ?]$$

$$threshold_1 := \boxed{\blacksquare \triangleright E, E, \square}$$

$$neuron_2 := [(\ ?, \ ?)(\ ?, \ ?)(\ ?, \ ?)(\ ?, \ ?)(\ ?, \ ?) \ll \square \gg \ ?, \ ?, \ ?, Z_2]$$

$$threshold_2 := \boxed{\blacksquare \triangleright E, E, \square}$$

The third neuron is the one on the left that $neuron_1$ and $neuron_2$ connect to.

$$neuron_3 := [(\ , \) (E, Y_1)(E, W_1)(I, Z_2)(\ , \) \ll \square \gg W_3, X_3, Y_3, Z_3]$$

$$threshold_1 := \boxed{\blacksquare \triangleright E, E, I, \square}$$

It is possible to generalize our model by allowing more input and output on each neuron. We can also augment the number of different types of neurotransmitters and even consider neurons that have more than one type of neurotransmitter. For a neuron, we can associate a series of threshold transformations that are different from each other to model changes in firing frequency. From a more general view, we can define transformations that change the connections, grow connections and change the threshold of some neurons.

7.10 Biology diagrams

Because of its flexibility, the language of transformations can create helpful biology diagrams to understand processes. This is useful for textbooks and

academic papers. As we have seen in the case of DNA, we can have a visual diagram and an inline notation (which is more computational). This reduces the gap between what is presented in diagrams and the formal mathematical process of the inline notation.

In the future we can think that there will be an automatic way to switch between 2D notation and inline notation. This will help understanding while keeping the computational power of the mathematical notation. It is possible that mathematicians and biologists will be encoding models and trying to refine them by switching between 2D and inline notations, expand a single form into its basic constituents and using powerful algorithms to expand our knowledge of biology.

7.11 3D Mesh transformations

3D computer graphics are built with a set of interconnected polygons. The polygons are connected with different angles in 3 dimensions and create the surface of an object. We can think of these polygons as the equivalent of digital pixels. With a large 3D database containing the mesh of 3D objects representing organs and body parts, we could transform the shape of 3D representation of body parts by using *mesh transformations*. Examples of mesh transformations on a triangular mesh are dividing triangles into structures composed of smaller triangles, changing the angle between triangles and dissolving triangles.

The reason for introducing mesh transformations is to reduce the gap between the way biological processes work and how we represent them as 3D computer graphics. For example, having a 3D model of a heart along with a set of transformations similar to the 2D heart presented above, we could modify the shape of the heart to adapt to different individuals. We could also define surgical mesh transformations to represent surgical incisions. Another way to think about some parts of our 2D heart model is to think that the contractions and relaxations of the cell were in fact mesh transformations. Interacting with 3D digital objects with mesh transformations allows us to not to leave our system of transformations that is used to represent biological processes.

7.12 Meta-medicine

In a large medicine database written in our language, we could recognize and abstract important biological structures and apply these to the understanding problems in other fields of medicine. Investigations such as these have been very fruitful in mathematics, where abstracting structures of the number systems gave rise to abstract algebra.

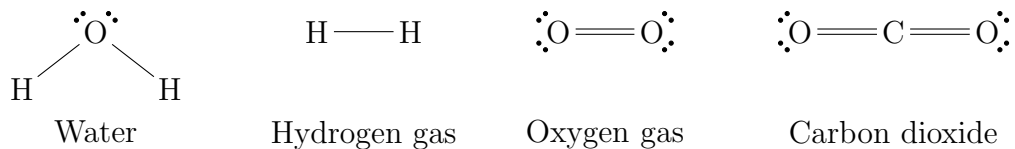
Also, the knowledge and abstract framework at different scale levels can be used at other scale levels. By extracting abstract principles observed in the human body we could use the same abstract structure to help understand the entire population. For example, if we consider RNA-messenger as information, we could try to understand based on biological processes how to improve knowledge dissemination during epidemics. One wonders if we can find new ways to deal with world health concerns as a whole by understanding how our own body functions.

8 Chemistry

We will now see that we can also use systems of transformations to represent chemical reactions. Note that we will use the arrow \mapsto instead of the classical notation \rightarrow for chemical equations.

8.1 Chemical reactions

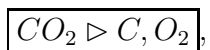
Examples of reactions are the combustion of carbon in oxygen $C + O_2 \mapsto CO_2$ and the combustion of hydrogen as used in space shuttles $2H_2 + O_2 \mapsto 2H_2O$. Water (H_2O), hydrogen gas (H_2), oxygen gas (O_2) and carbon dioxide (CO_2) are written as Lewis structures as follows.



We can write the carbon combustion reaction in the form of a Lewis structure as follows.



As a coarse transformation, we can write carbon combustion as



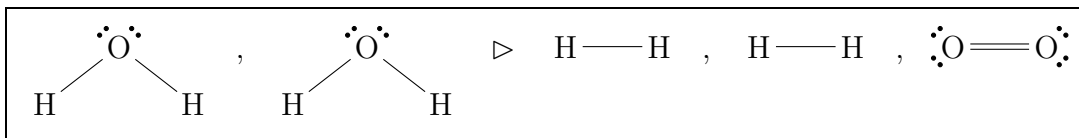
and we can write hydrogen combustion as



These can also be written in Lewis notation as

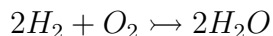


and

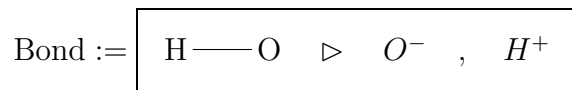
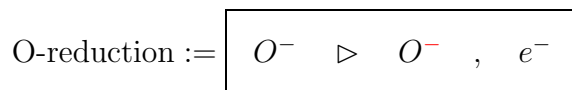
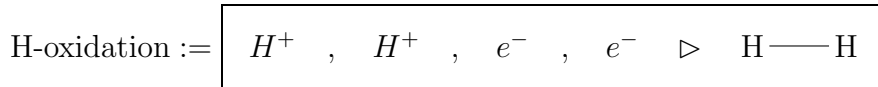


8.2 Studying hydrogen combustion

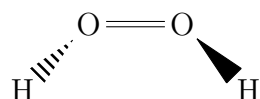
One way to write a more precise model of the reaction



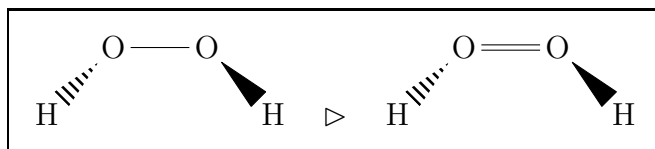
which represent hydrogen combustion in oxygen is by using the following transformations. When hydrogen is burned in the presence of oxygen the product is water. From here we can assume that one of the paths goes through hydrogen peroxide.



By applying these three transformations in parallel to $2H_2 + O_2$, the two H_2 will become two H^+ and two free electrons because of H-oxidation. Then O-reduction will give an extra electron to the two oxygens. This will result in the temporary molecule

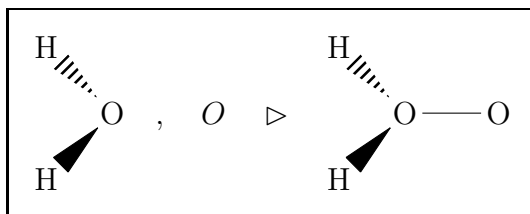


Now, when each oxygen shares a covalent bond with a hydrogen atom, a bond is removed between the oxygens to give a hydrogen peroxide molecule (H_2O_2). This is represented by



Note that the black triangle is a classical notation to indicate that the HO bond is oriented towards the viewer and the dashed triangle means that the bond is oriented away from the viewer.

Now, since the oxygens do not have a negative superscript anymore, thus the transformation O-reduction can be applied again to each O. This means that the transformation *Bond* will be applied again to one of the oxygens. We now only need to dissolve the second bond between the oxygen atoms with the following transformation.



In the present case, we dissolved the bond when there were three attached H atoms instead of four, this is probably more likely. Finally, OH^- hydroxide ion will react with H through the transformation *Bond* to form the second H_2O molecule. This is one of the many paths to go from $H_2 + O_2$ to H_2O and when hydrogen is burned, it is likely that all the paths are followed in different proportions.

All these possible pathways could be computationally studied to understand the inner working of a reaction.

8.3 Reaction mechanisms

We will now give an example of a representation of a reaction mechanism.

The decomposition of nitrogen dioxide into nitric oxide and oxygen is an example of a reaction mechanism. The net balanced equation is $2NO_2 \rightarrow 2NO + O_2$.

The mechanism of this reaction is believed to follow these steps.

1. $2NO_2 \rightarrow NO_3 + NO$
2. $NO_3 \rightarrow NO + O_2$

This can be represented as a system of transformations as

$$[\boxed{NO, O_2 \triangleright NO_3}, \boxed{NO_3, NO \triangleright NO_2, NO_2}] \rightarrow \{NO_2, NO_2\}.$$

9 Absolute versus relative transformations

Until now we were mostly interested in transformations from a global perspective. To model the evolution of a system, an absolute observer defines transformations which explain the changes observed in the system. We mostly defined transformations that somewhat stand out of the system and are then applied in the system when the conditions are right. The transformations defined as such are only based on our evaluation of the system and are used to describe what we see. From the point of view of a molecule, there is no global law that comes to affect the molecule. It is the same for biological processes, there is nowhere in the body where there is a set of transformations waiting to modify parts of the body.

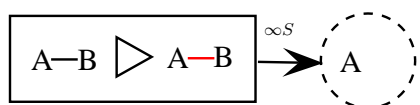
As we have seen, the absolute transformations can model many things and follow the way we understand and describe the world. This leads us to the concept of relative transformations. It is not a really big change in the notation, but mostly a change in perspective. This perspective is closely related to the idea that until a transformation is applied we cannot detect it. We will now give two examples of systems that can be said to be relative.

9.1 Relative molecules

Instead of defining an initial form containing different elements and a collection of transformations applied to it, we will define elements that come

with their transformation. We have already seen in section 7.9.1 that we can define neurons that come with their respective threshold transformation. This is an example of a form which is composed of a certain structure and a transformation where we are closer to the relative view.

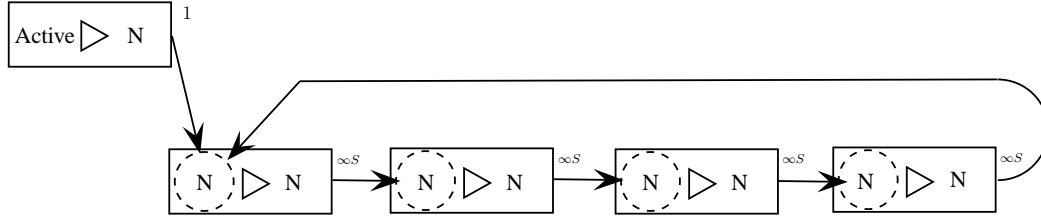
We will now look at another example which is related to how molecules behave in a laboratory jar. To do this we take an initial form and put an atom A in it. Assume that if atom A has no bond and meets an atom B it will create a bond with the atom B . This is represented as follows.



This defines an atom A along with its property to create a bond with an atom B . In a solution, this atom will move and when an atom B enters in its initial form, then a bond will be created with the atom B . As indicated by the bond in red in the cause of the transformation, the bond will be created only if the atom A does not already have a bond. The initial set is understood to have a size and is moving along with the atom A . The atom A meeting an atom B will result in $A - B$ in the initial form with the transformation still pointing to the initial form. Note that by writing \sharp instead of ∞ will make the transformation disappear. This can be understood as a relative model where each form has as a certain range of interactions given by the size of the initial form. Eventually, each atom or molecule could be defined as a collection of forms and transformations. We could also extend approach in a way that the transformations themselves also react to each other.

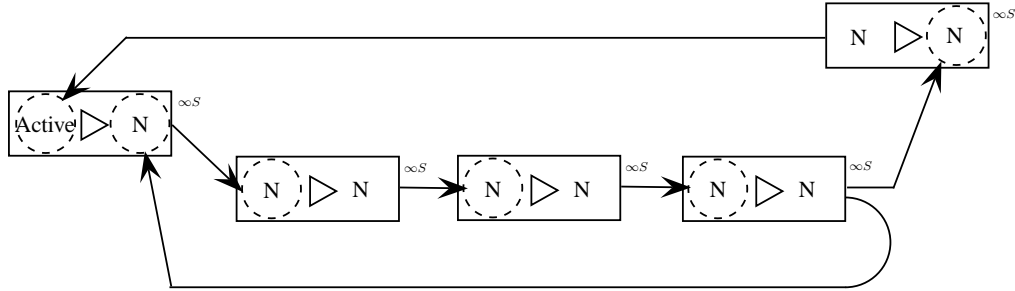
9.2 Chains of transformations

With the use of higher order transformation as defined in section 4.9, we can define the following chain of transformations. The N 's in the initial forms can be viewed as the initial state of the cell $\triangleright N$. Changing the state of the first cell to *Active* will induce a domino effect and change the state of the other cells to active. Note that since it is not needed in the following model, we have not written a number over the arrow to indicate the order in which they should be applied.



Interestingly, changing the first cell back to N with the transformation $N \triangleright Active$ will change all the other cells to N . This abstract example can be interpreted as a solid-state drive that keeps the data N or $Active$.

The following example of a chain, will change the state of the three lower cells to $Active$ and when it is done, it will automatically change it back to N . This abstract model could be viewed as a neuron firing or muscle cells contracting and relaxing.

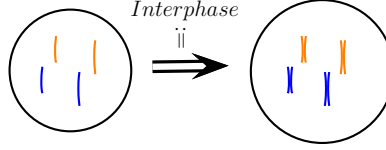


10 Computing treatments and processes

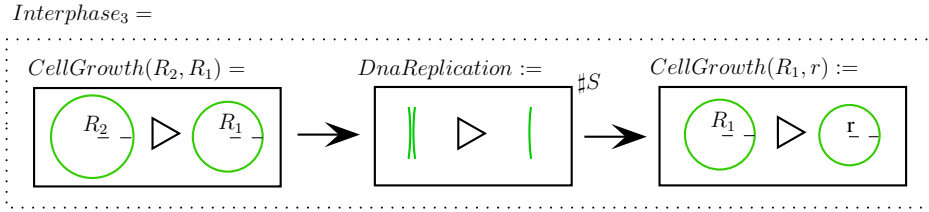
Using our language, we can mathematically and visually represent biological systems. This is useful to communicate and find new lines of investigations to finding remedies. We can write questions as equations and solve the equations. The ultimate goal is to compute treatments and unknown processes by solving equations.

10.1 Solving for processes

Earlier in section 7.1, we coarsely defined interphase as follows.



In section 7.2, we refined our interphase model as



where the right-hand side will be defined a $\{Cell\}$. In section 7.3 we refined the *DnaReplication* collection of transformations.

We now give a simple example of an equation where the solution is a process. We take the following equation where X is the only variable.

$$CellGrowth(R_2, R_1) \rightarrow X \rightarrow CellGrowth(R_1, r) \rightarrow \{Cell\} \asymp Interphase \rightarrow \{Cell\}$$

Here, recall that the symbol \asymp indicates that each side reduces to the same form.

We can also write this as follows with the understanding that the subscript indicates that both are over the initial form *Cell*.

$$CellGrowth(R_2, R_1) \rightarrow X \rightarrow CellGrowth(R_1, r) \asymp_{\{Cell\}} Interphase.$$

We already know that one solution for this equation is $X = DnaReplication$, but if we were given such an equation, we could see by inspecting *Interphase* and the *CellGrowth* that we would only need to copy the chromosomes to satisfy the equation. This is a simple example of solving an equation to find a process. When we are solving equations we can ask for a solution to be coarse or fine. For example, we could ask for X to contain only series of transformations on nucleotides, but this would imply that we have to write the DNA composed of nucleotides and not just represented by orange lines.

10.2 Solving for antiviral drugs

A reproducing virus can be represented at the coarse level by

$$\boxed{virus, virus \triangleright virus} \rightarrow \{virus\}.$$

An antiviral drug X would be a molecule that changes a virus in a way that it stops replicating in the host cell. We can model this by the following system of equations.

This first equation modifies the virus.

$$\boxed{X \triangleright virus} \rightarrow \{virus\} \Rightarrow \{virusModified\}$$

The next two equations indicates that the modified virus cannot replicate into a virus or modified virus.

$$\begin{array}{c} \boxed{virusModified, virusModified \triangleright virusModified} \rightarrow \{virusModified\} \\ \Downarrow \\ \{virusModified, virusModified\} \end{array}$$

$$\boxed{virusModified, virus \triangleright virusModified} \rightarrow \{virusModified\} \not\Rightarrow \{virusModified, virus\}$$

The fourth equation concerns the efficiency of the drug and says that a modified virus should not lose its modification.

$$\boxed{virus \triangleright virusModified} \rightarrow \{virusModified\} \not\Rightarrow \{virus\}$$

The fifth equation indicates that the drug should not interfere with the host cell.

$$\boxed{X \triangleright host} \rightarrow \{host\} \not\Rightarrow \{unhealthyHost\}$$

Note that we have used the notation ‘ $\not\Rightarrow$ ’ to indicate that there was no reduction.

This was a simple example, but the setting would be the same for a much more refined model. With a refined model of a virus at the protein or molecular level including many of its processes, a computer would be able to compute new drugs based on such systems of equations.

10.3 Understanding cancer

Let's define a cell along with a marker $\langle Cell, marker \rangle$ where 'marker' can take the values *quiet* and *ready*. The process of cell division follows the sequence of transformations below.

$$\boxed{\langle Cell, quiet \rangle, \langle Cell, quiet \rangle \triangleright \langle Cell, quiet \rangle} \rightarrow \boxed{ready \triangleright quiet} \rightarrow \{\langle Cell, quiet \rangle\}$$

The transformation $\boxed{ready \triangleright quiet}$ is what initiates the cell division cycle when needed. When the cycle is initiated, this will reduce to the two quiet cells $\{\langle Cell, quiet \rangle, \langle Cell, quiet \rangle\}$.

One way to model something that causes cancer and a cancerous cell is with the two following equations where X is an unknown transformation.

$$X \rightarrow \{\langle Cell, quiet \rangle\} \Rightarrow \{\langle CancerCell, ready \rangle\}$$

$$\boxed{\langle CancerCell, ready \rangle, \langle CancerCell, quiet \rangle \triangleright \langle CancerCell, quiet \rangle}$$

Note that one of the cancerous cell is staying in the ready state without the cycle being initiated. A more aggressive form of cancer would be

$$\boxed{\langle CancerCell, ready \rangle, \langle CancerCell, ready \rangle \triangleright \langle CancerCell, quiet \rangle}$$

To refine this model further we need define how the immune system reacts to the cancerous cells by defining the transformations *Immune* which can only apply to cells as follows.

$$Immune \rightarrow \{\langle Cell, quiet \rangle\} \Rightarrow \{\langle Cell, quiet \rangle\}$$

$$Immune \rightarrow \{\langle Cell, ready \rangle\} \Rightarrow \{\langle Cell, ready \rangle\}$$

$$Immune \rightarrow \{\langle UnhealthyCell, quiet \rangle\} \Rightarrow \{ \quad \}$$

$$Immune \rightarrow \{\langle UnhealthyCell, ready \rangle\} \Rightarrow \{ \quad \}$$

For the cancerous cell not to be removed, the immune system would be leaving many cancerous cells untouched and we would have to include the following in a definition of cancer.

$$Immune \rightarrow \{\langle CancerCell, ready \rangle\} \Rightarrow \{\langle CancerCell, ready \rangle\}$$

10.4 Calculating mathematical solutions

Now that we can write questions from the domain of medicine as transformation equations, we have to develop ways to solve them.

An abstract example is the equation

$$\boxed{b \triangleright c} \rightarrow \boxed{X \triangleright a} \rightarrow \{a\} = ab.$$

By inspection and trial, a solution is $X = ac$. But there is a computational way to do it. When we are solving equations such as $2 + x = 5$, we can subtract on both sides to get $-2 + 2 + x = -2 + 5$ which gives $0 + x = -2 + 5$ and a final solution $x = 3$.

For us, the inverse of $\boxed{c \triangleright b}^{\#S}$ will be $\boxed{b \triangleright c}^{\#S}$, since we have that

$$\boxed{b \triangleright c}^{\#S} \rightarrow \boxed{c \triangleright b}^{\#S} \rightarrow \{Y\}$$

reduces to $\{Y\}$ when Y is any string of letters. If we want to solve the following equation, we can use the inverse to isolate X on one side.

$$\begin{array}{rclcl} \boxed{c \triangleright b}^{\#S} \rightarrow \boxed{b \triangleright c}^{\#S} \rightarrow \boxed{X \triangleright a}^{\#S} \rightarrow \{a\} & = & \{ab\} \\ \boxed{c \triangleright b}^{\#S} \rightarrow \boxed{b \triangleright c}^{\#S} \rightarrow \boxed{X \triangleright a}^{\#S} \rightarrow \{a\} & = & \boxed{c \triangleright b}^{\#S} \rightarrow ab \\ \boxed{X \triangleright a}^{\#S} \rightarrow \{a\} & = & \{ac\} \\ \{X\} & = & \{ac\} \\ X & = & ac \end{array}$$

10.5 Producing the remedies

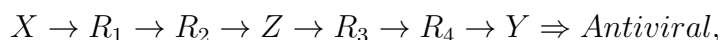
If we know that a certain molecular structure will have a wanted effect on a system, for example an antiviral drug, we can test this drug in a large and precise model of the body to see where this drug acts and discover the potential side effects.

If we have a satisfying drug, one problem is to be able to synthesize this molecule *Antiviral*. This can be done by solving the equation for X and Y where X is composed of transformations at the molecular level.

$$X \rightarrow Y \Rightarrow \textit{Antiviral}$$

In this case, Y would be the starting compound and X would be a series of transformations. At first the set of allowable transformations for X would

be from a database of understood reactions. If we cannot synthesize the antiviral drug with known reactions, we can focus on a step of the series for which we don't know the reaction. Suppose that we know the reactions R_1, R_2, R_3 and R_4 and that



then we would need to focus on the unknown reaction Z .

A useful tool would be to define a mathematical measure of closeness between molecules. This would help us evaluate which molecule out of two is closer under that measure to another molecule. For example, if we want to reach X from Y , we could calculate a value of 'molecular distance' D between the molecule resulting from the reaction $R_4 \rightarrow Y$ and X , and between $R'_4 \rightarrow Y$ and X . If $D(R_4, X) < D(R'_4, X)$, we will choose R_4 as the first transformation applied to Y .

11 Computing techniques

We will now briefly discuss some potential areas of investigation related to computing the solutions of a system of equations.

11.1 Solving equations

In mathematics, there are many techniques and algorithms to solve different types of equations. For example, algebra (and other methods) permits us to solve equations involving polynomials. We have seen simple examples of a technique to solve a system of transformations in section 10.4, but there are many more techniques that would need to be developed to handle more complex systems involving transformations.

Linear algebra equations and matrices were initially developed to solve systems of linear equations. It would be interesting to develop objects which would themselves be forms that can be applied to systems of equations of transformations. These objects could act in a way that is similar to matrices and help solve certain types of systems.

11.2 Principle of resilience and propagation

Most processes and features we observe in the body are resilient. This means that errors, mutations and modified processes are naturally eliminated from

the body. We could use this as a way to compute unknown processes. When investigating a hypothetical process, we can evaluate the resilience of a process to determine if it is stable in the body and could have survived. We can probe and test the process by introducing errors in the flow of steps and see if the process destabilizes or not. If the process destabilizes with small interactions from the body itself or the environment, then this hypothetical process must be discarded. This is a very useful principle from a computational point of view. To solve a problem or find a representation of a biological process, the computer could generate many different processes and eliminate them by testing if they are resilient.

For example, in a certain series of steps, if we know that something is produced at one of the steps, but do not know how it was produced, the computer could generate different ways (using what is present in the cell or not) and then test the resilience of each process. Each process could also be tested for how they propagate in the system. If it implies a buildup somewhere else and it is never observed in reality, this process could be discarded. If the process disturbs the resilience of another process, thus making the other process disappear, then the process could also be discarded. A classical algorithm which might be useful here would be a genetic algorithm.

11.3 Optimization

Usually biological mechanisms are very well optimized to accomplish their task. Considering the complexity of the brain and DNA replication, we see they are the result of very efficient processes. Processes that we want to discover can often be close to an optimal solution. This can help us discard many possible solutions by considering measures associated with it such as speed, polyvalence, number of steps, stability and precision.

When we have a general framework of a process, there are many unknown constants that need to be determined in order to obtain an appropriate model. If there is a finite number of unknowns, this model can be optimized by entering different values and evaluated to see how the model will perform. The choice of values can be done by using a genetic algorithm or quantum computers using annealing algorithms.

11.4 Main algorithm

Below is a sketch of an algorithm to find new treatments and biological processes with a computer. Eventually, this main algorithm should be developed and refined.

1. Write the equations representing your question.
2. Access a database of known processes, transformations and molecules. Including quantities and proportions.
3. Separate the problem into almost disjoint parts or clusters.
4. Try solving the equation directly using direct techniques such as inverse transformations.
5. Use known computing techniques and algorithms. For example, algorithms based on resilience and optimization.
6. Introduce a molecule or a process that is not usually present in the body to try solving the equations.
7. Evaluate a solution to see if there are unwanted side-effects and interactions with biological mechanism.

11.5 Computable and uncomputable functions

At some point we will meet transformation equations that are uncomputable or out of reach. An idea would be to define an object that is a solution to the transformation equation and build a mathematical theory based on this definition. The generic example behind this is trying to solve the polynomial $x^2 = -1$ over the real numbers. Since squaring any real number will result in a positive number we see that this equation has no solution over the real numbers. Euler's idea was to define an object i such that if you square it you get -1 . This way, we have that i is a solution to the equation $x^2 = -1$. Mathematicians did not stop there, using this; they constructed the set of complex numbers along with its algebra and amazing properties. In a way, i and complex numbers could be seen as a 'programming oracle' with an algebraic structure. A similar approach for transformations could be beneficial in investigating the world of uncomputable functions, transformations and related structures.

12 Computer Science

12.1 Programming statements

Programming statements can be viewed as the atoms or sentences of programming. We will now demonstrate that we can also represent programming statements in our language.

12.1.1 If-statement

We can represent an if-statement as follows. Example of conditions are $x = 5$, $x < 10$ and $b = \text{true}$. We will say more about conditions in the section about the for-loop.

$$\boxed{B \triangleright A} \rightarrow \boxed{\text{condition}, A \triangleright \text{condition}, A} \rightarrow \{\text{condition}, A\}$$

If the condition is satisfied, then the identity transformation which rewrites *condition*, A by *condition*, A is applied and thus permits the next transformation in the sequence to change A or perform an operation depending on what A is.

We can represent an if-else-statement as follows.

$$\begin{aligned} & \boxed{B \triangleright A} \rightarrow \boxed{\text{condition}, A \triangleright \text{condition}, A}, \\ & \boxed{C \triangleright A} \rightarrow \boxed{\text{condition}, A \triangleright \text{condition}, A} \end{aligned} \rightarrow \{\text{condition}, A\}$$

Thus, since *condition* is written in red, it means that the transformation applies when the condition is not satisfied, only one of these sequences of two transformations will affect A . If we want all the transformations to disappear when one of the sequences of transformation has been applied, we can use the sharp symbol \sharp as follows.

$$\begin{aligned} & \boxed{B \triangleright A} \rightarrow \boxed{\text{condition}, A \triangleright \text{condition}, A}, \\ & \boxed{C \triangleright A} \rightarrow \boxed{\text{condition}, A \triangleright \text{condition}, A} \end{aligned} \sharp \rightarrow \{\text{condition}, A\}$$

12.1.2 Switch-statement

A switch statement is done in a similar way than an if-then-else statement, but with the use of multiple cases. A general example for three cases is

written as

$$\begin{aligned} & (\boxed{B \triangleright A} \rightarrow \boxed{\text{case1}, A \triangleright \text{case1}, A}, \\ & \boxed{C \triangleright A} \rightarrow \boxed{\text{case2}, A \triangleright \text{case2}, A}, \\ & \boxed{D \triangleright A} \rightarrow \boxed{\text{case3}, A \triangleright \text{case3}, A}) \rightarrow \{\text{condition}, A\}. \end{aligned}$$

12.1.3 For-loop

We present two ways to do a for-loop. The first is the simplest.

$$\boxed{A' \triangleright A}^n \rightarrow \{A\}$$

For $n = 4$, this will return A''' .

In a classical for-loop, we need an increment i and a condition based on the increment, for example $i < 4$. The increment will be a natural number. To initiate the loop, we need to first define $i := 1$.

$$[\boxed{i := i + 1, A' \triangleright i, A} \rightarrow \boxed{i \leq 4, A \triangleright i \leq 4, A}]^\# \rightarrow \{i, A\}$$

Note that in the transformation $\boxed{i := i + 1, A' \triangleright i, A}$ we are giving another value to i , this can be viewed as a new feature of our language. Concerning the condition $i \leq 4$, we could apply some sequence of transformations to make the symbols $i \leq 4$ appear in the set before applying the transformation $\boxed{i \leq 4, A \triangleright i \leq 4, A}$, but there is another way. We can define $\boxed{i \leq 4, A \triangleright i \leq 4, A}$ as a collection of parallel transformations as follows.

$$\boxed{i \leq 4, A \triangleright i \leq 4, A} := [\boxed{1, A \triangleright 1, A}, \boxed{2, A \triangleright 2, A}, \boxed{3, A \triangleright 3, A}, \boxed{4, A \triangleright 4, A}]^\#$$

Thus, when one of the transformations is applied, $\boxed{i \leq 4, A \triangleright i \leq 4, A}$ will disappear and let the transformation $\boxed{i := i + 1, A' \triangleright i, A}$ from the for-loop to be applied.

12.1.4 While-loop

The while-loop will check the condition and then execute the content until the condition is not satisfied.

$$[\boxed{A' \triangleright A}] \rightarrow [\boxed{\text{condition}, A \triangleright \text{condition}, A}, \boxed{\phantom{\text{condition}}, A \triangleright \text{condition}, A}]^{\#}]^{\infty}$$

$$\downarrow$$

$$\{\text{condition}, A\}$$

12.1.5 Do-loop

The do-loop will execute the content and then check the condition to see if it should continue.

$$[[\boxed{\text{condition}, A \triangleright \text{condition}, A}, \boxed{\phantom{\text{condition}}, A \triangleright \text{condition}, A}]^{\#}] \rightarrow \boxed{A' \triangleright A}]^{\infty}$$

$$\downarrow$$

$$\{\text{condition}, A\}$$

12.2 Forms programming

We have already seen in section 4.3.2 and 4.3.3 that we can name any form with added variables. This permits us to use object programming techniques in our language. Since we can build programming statements with transformations, this means that after building a scientific model in the language of transformations, we can build programs without leaving the language. This points toward a new and powerful programming language based on forms and transformations.

If such a programming language is developed based on modern higher programming languages, one has to make extensive use of arrays and sub-arrays. For example, an inline transformation can be viewed as an array of two components where each component contains an array of n components. For transformations in diagram form, a transformation would be viewed as an array of two components where each component contains a m by n two-dimensional array. Similarly for a 3D representation, one would use transformation components that are three-dimensional arrays.

Interestingly, this array-based approach to transformations points towards what we could understand as n -dimensional transformations and forms. This means that the language of transformations could be extended to higher dimensions.

12.3 Conway's game of life

We will now give a model of Conway's game of life. Conway's game of life is one example of a wider class of objects called cellular automata.

It is understood that the initial set is a 3 by 3 grid along with the live cell in it. The following transformation defines Conway's game of life. For a larger grid, this collection of transformations is applied to each 3 by 3 grid of the larger grid.

$$\left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}, \text{red square} \right\} \triangleright \left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & \text{green square} & \\ \hline & & \\ \hline \end{array}, \text{red square} \right\}$$

$$\left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{red square} \right\} \triangleright \left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & \text{green square} & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{red square} \right\}$$

$$\left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & \text{green square} & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{red square} \right\} \triangleright \left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & \text{green square} & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{red square} \right\}$$

$$\left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & \text{green square} & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{black square}, \text{red square} \right\} \triangleright \left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & \text{green square} & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{black square}, \text{red square} \right\}$$

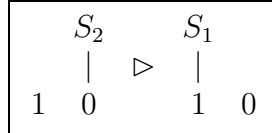
$$\left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{black square}, \text{black square} \right\} \triangleright \left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & \text{green square} & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{black square}, \text{black square} \right\}$$

$$\left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{black square}, \text{red square} \right\} \triangleright \left\{ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}, \text{black square}, \text{black square}, \text{black square}, \text{red square} \right\}$$

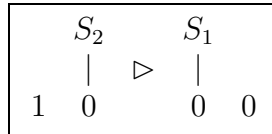
Interestingly, this opens up a lot of possibilities of cellular automata based on Conway's game of life. For example, the sub-grid does not have to be 3 by 3 but can have an irregular shape, we can have much more complex rules or the grid can take the form of another type of tiling of the plane. We are also not only restricted to the plane, but could define something similar for the surface of a sphere.

12.4 Turing machines

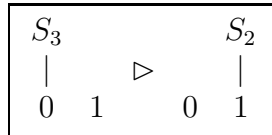
We can also define Turing machines by defining transformations similar to the one below. In state S_1 read 1, write 1 and move one step to the right and go into state S_2 .



In state S_1 read 0, write 1 and move one step to the right and go into state S_2 .



In state S_2 read 1, write 1 and move one step to the left and go into state S_3 .



Interestingly, we now have no restriction on what the symbols or the tape can be. The tape could even take the form of a tree or a fractal. Moreover, we could take advantage of the capability for transformations to affect themselves or other transformations to generalize Turing machines.

13 Mathematics

13.1 Functions

Functions are essential objects of modern mathematics and can also be described with transformations. A function f from X to Y associates to each element of the set X an element of the set Y . One way to define a function with a transformation is as follows where y_x is the element of Y which is associated to x . This is basically the same as defining a collection of ordered pairs to define a function.

$$f(x) := \boxed{y_x \triangleright x}$$

If we want to define the polynomial function $f(x) = x^2$, we can write this as follows based on the multiplication defined in section 4.3.3.

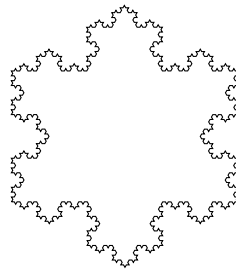
$$f(x) := \boxed{x \times x \triangleright x}$$

Transformations are more flexible than functions or more generally, sets of ordered pairs. Like sets of ordered pairs, transformations can point to two objects and don't need to apply to all elements. Unlike functions, we don't need to define the image set, transformations can also be applied to other transformations and we can have multiple occurrences of an element in a initial set. Usually, when working with functions, the domain and the image set of the function are known beforehand and the elements are of the same type. This is not required for transformations. If we look at an initial set and the reduced set, we can create a set of ordered pairs based on the transformation.

13.2 Fractals

By construction, the language of transformations can naturally define fractals. We will now look at and define two well-known fractals, the Koch and Sierpinsky fractals.

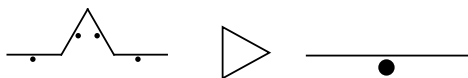
At first thought, the Koch snowflake



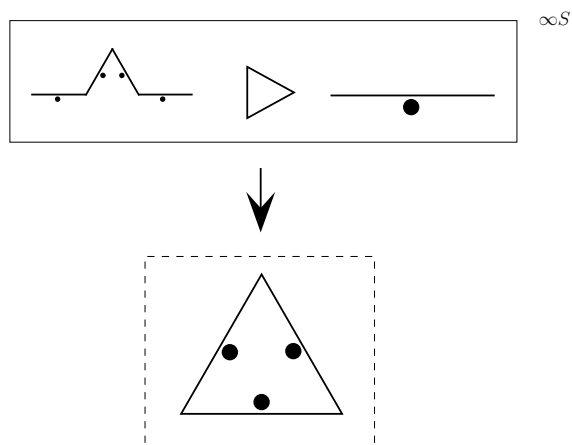
could be defined by the following transformation.



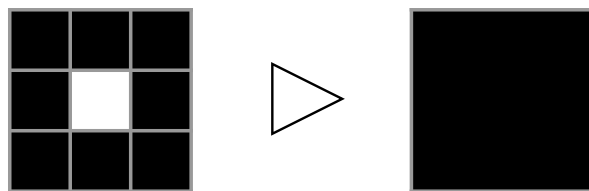
But, one problem with the above transformation is that there is nothing that is forcing the snowflake to be oriented outside. Orientation can be achieved by using a dot on one side of the line.



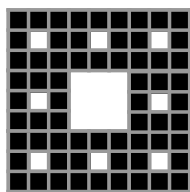
Applying this transformation to a triangle will result in the Koch snowflake.



For the Sierpinsky carpet, the following transformation (which we will call the Sierpinsky transformation) will be used.



If we applied the Sierpinsky transformation to a black square nine times we get the following shape.



The Sierpinsky carpet is defined by applying the Sierpinsky transformation an infinite number of times to a black square.

Again, this opens many possibilities of modified Koch or Sierpinsky fractals. More interestingly, this indicates that the language of transformation could be a natural mathematical choice to develop a calculus for fractals.

13.3 Differential calculus

Although we will not present it here, it is possible to define division and the entire number system with the use of transformations. Assuming that we have division, we can define the derivative of a function as follows.

$$f'(x) := \left[\frac{h}{2} \triangleright h \right]^{\infty S} \rightarrow \frac{f(x+h) - f(x)}{h}$$

Note, that the infinite sequence $\left[\frac{h}{2} \triangleright h \right]^{\infty S}$ acts like the limit when h tends towards 0.

13.4 Meta-mathematics

If we translate two mathematical theories T_1 and T_2 into the language of transformations, we can investigate how close two such theories are to each other. For this we can write

$$\boxed{X \triangleright T_1} \rightarrow T_1 \asymp \boxed{Y \triangleright T_2} \rightarrow T_2.$$

This means that under the transformations X and Y the theories will reduce to the same form. Depending on what X, Y are and what the theories reduce to, we could evaluate how close the theories are to one other or how they relate. This is similar to what is done in category theory where functors between categories and comparisons between specific internal structures of categories give a way to compare and classify the categories. One advantage with transformations is that we have a means to relax some restrictions required by categorical structures and thus generalize the tools of category theory.

One of the main activity, and maybe the only activities in Mathematics is to classify abstract objects. In essence, objects are analysed with different

tools and then classified as the same or different from other objects. If objects are different, a degree or level of difference can be associated. Here are a few examples spanning basic and advanced mathematics.

A key element of arithmetic is equality. $1 + 2 = 3$ can be seen as stating that $1 + 2$ is the same as 3. Commutativity $xy = yx$ is also an expression of sameness classifying the structures xy and yx . The expression $2x = 1$ is asking if there is a way to have $2x$ to be the same as 1. This is not possible for integers, but defining a new number $\frac{1}{2}$ forces this expression to be valid. Also, $30 = n \times 15$ can be viewed as asking how far is 15 is from 30.

Classification is done at all levels of mathematics, other examples are homology, category theory and topology. An interesting question is to ask if mathematics can solely be viewed as the activity of classification along with the introduction of new definitions and objects.

The advantage of the language of transformation is that all mathematical objects can be constructed with forms and this allows us to compare all concepts and objects with each other. A homogeneous language might be the key to build a large map of the mathematical and scientific landscape.

13.5 Continuous transformations

The language that was presented here (like classical logic and set theory) has a discrete flavor. Since we can represent an infinite number of transformations, as seen in section 13.3, we can approach infinitely small elements. We have seen that we can use probabilities on transformations and we can wonder if there is a way to make the application of transformations less discrete. For us, there is before and after the application of a transformation, but we could eventually define a way to continuously apply a transformation where there is a continuum ranging from not applied to applied. We would eventually have to describe the real number with transformations.

13.6 Formal description of the language

We have presented the language of transformations in a progressive way with the aim to make it easy to understand. Although we will not be doing it here, it is possible to give a formal description of the language by using axioms and definitions.

14 Physics

We will now present simple models in physics.

14.1 Motion

For the physics of moving bodies, we will consider that there is a minimal unit of length u_d that an object can move and that there is a minimal unit of time u_t . Since they are arbitrary values, they can always be adjusted based on our measuring tools.

We describe an object moving in a straight line along with its associated clock as follows.

$$v_{1/1} := \boxed{\dashrightarrow \bullet, \rightsquigarrow \top \quad \triangleright \quad \bullet, \top}$$

When applied to the initial set $\{\bullet, \top\}$, this transformation can be interpreted as giving the speed 1 unit of distance per unit of time to the particle depicted by ‘ \bullet ’. Here the dashed arrow \dashrightarrow indicates that one unit was covered and the arrow \rightsquigarrow indicates that one unit of time elapsed.

The following transformation defines a speed of 2 units of distance per unit of time ($2u_d/u_t$).

$$v_{2/1} := \boxed{\dashrightarrow \dashrightarrow \bullet, \rightsquigarrow \top \quad \triangleright \quad \bullet, \top}$$

If we apply this speed $v_{2/1}$ repeatedly to $\{\bullet, \top\}$ our particle is seen as having an acceleration of 0.

We can define uniform acceleration of 1 by using the following series of five speed transformations.

$$a_{1/1} := v_{5/1} \rightarrow v_{4/1} \rightarrow v_{3/1} \rightarrow v_{2/1} \rightarrow v_{1/1}$$

Although much more work is needed to develop this approach, transformations can bring a new way to understand physics. An interesting property of this approach is that these transformations for speed and acceleration do not need an absolute coordinate system to be expressed, they can be seen as being relative to the moving body.

14.2 Attractive and repulsive forces

We now present a model of an attractive and repulsive force. The goal is to show that transformations can provide a new way to look at the fundamental forces.

Let's define an object (x, E) where x is the length of the last distance covered and E can be seen to represent its internal energy or mass. If x is negative, it means that it covered the distance from right to left and if it is positive, it covered the distance from left to right. We will consider the number 1 written between square brackets as $[1]$ to represent space without the presence of objects, this can be viewed as natural space. If the number is different, it will mean that the space is stretched or contracted. For our model, we put an object $(0, M)$ at rest of mass M where M is larger than 1. Thus we have the following.

$$[1](0, M)[1][1][1][1][1][1][1]$$

In our model, the mass will change the size of space elements. If the mass is larger than 1 then the surrounding space will stretch and if the mass is smaller than 1 then the surrounding space will contract. The closer the space to the object, the larger the stretch or contraction depending on our model. For example, a mass M larger than 1 will have a stretching effect on its surrounding space.

$$\left[1 + \frac{M-1}{2}\right](0, M)\left[1 + \frac{M-1}{2}\right]\left[1 + \frac{M-1}{3}\right]\left[1 + \frac{M-1}{4}\right]\left[1 + \frac{M-1}{5}\right]\left[1 + \frac{M-1}{6}\right].$$

Note that because of the square brackets $\left[\frac{M-1}{2}\right]$ must be viewed as a distance and not a mass. Our choice in the amount of space stretching is arbitrary and should be adapted with a more precise sequence to approach a realistic physical model.

If we place an object of mass $m > 1$ on the right along with more space on the right of it, such that m is negligible compared to M , we get

$$\left[1 + \frac{M-1}{2}\right](0, M)\left[1 + \frac{M-1}{2}\right]\left[1 + \frac{M-1}{3}\right]\left[1 + \frac{M-1}{4}\right]\left[1 + \frac{M-1}{5}\right](0, m)\left[1 + \frac{M-1}{6}\right].$$

The following transformations can be seen as a force acting on the mass m . If $u > v$ we apply

$$\boxed{(-u, m)[u][v] \triangleright [u](x, m)[v]}.$$

If $v > u$, then we apply

$$\boxed{[u][v](+v, m) \triangleright [u](x, m)[v]}.$$

These transformations can be interpreted to say that a mass will move in the direction where there is the most space available and will record the distance it just covered.

By applying these transformations to

$$\left[1 + \frac{M-1}{2}\right](0, M) \left[1 + \frac{M-1}{2}\right] \left[1 + \frac{M-1}{3}\right] \left[1 + \frac{M-1}{4}\right] \left[1 + \frac{M-1}{5}\right] \left[1 + \frac{M-1}{6}\right],$$

we find that the mass m will cover more and more distance, thus accelerating toward the mass M . The distance it just covered can be viewed as its speed or kinetic energy.

Now let's take a mass m_e smaller than 1, this mass could be understood as the mass of an electron. If we construct its surrounding space in a similar manner as the mass M , we have that $m_e - 1$ is a negative number. This means that the space around the mass m_e is contracting.

$$\left[1 + \frac{m_e-1}{2}\right](0, m_e) \left[1 + \frac{m_e-1}{2}\right] \left[1 + \frac{m_e-1}{3}\right] \left[1 + \frac{m_e-1}{4}\right] \left[1 + \frac{m_e-1}{5}\right] \left[1 + \frac{m_e-1}{6}\right].$$

If we put a test particle of mass smaller than m_e and negligible compared to m_e close to the electron, we have

$$\left[1 + \frac{m_e-1}{2}\right](0, m_e) \left[1 + \frac{m_e-1}{2}\right](0, m_t) \left[1 + \frac{m_e-1}{3}\right] \left[1 + \frac{m_e-1}{4}\right] \left[1 + \frac{m_e-1}{5}\right] \left[1 + \frac{m_e-1}{6}\right].$$

Since there is more space on the right, the test particle will move towards the right after applying the transformation force.

With two electrons we will have repulsion since each will act by repulsing the other. But if we have a mass M larger than 1 and larger than the difference between m_e and 1, then the electron will repulse the large mass a very small amount, but the large mass will attract the electron much more.

From this point of view, we have one force which expressed as attractive for a large mass and repulsive for light particles such as electrons. In this simple model, we have to consider that protons have a mass larger than 1.

A variant of our transformation force is to require for $x < 0$ that we apply

$$\boxed{(-u, m)[u][v] \triangleright [u](x, E)[v]},$$

when $u > v + xE$ and

$$\boxed{[u][v](+v, m) \triangleright [u](x, E)[v]}.$$

when if $v + xE > u$.

If $x > 0$ then we use

$$\boxed{(-u, m)[u][v] \triangleright [u](x, E)[v]},$$

when $u < v + xE$ and

$$\boxed{[u][v](+v, m) \triangleright [u](x, E)[v]}.$$

when if $v + xE > u$.

Finally if $x = 0$, we use

$$\boxed{(-u, m)[u][v] \triangleright [u](0, E)[v]},$$

when $u > v$ and

$$\boxed{[u][v](+v, m) \triangleright [u](0, E)[v]}.$$

when if $v > u$.

With these, the object (x, E) will accelerate towards a larger mass, pass it and continue on its way until it is stopped and goes back towards the mass. This can be viewed as an oscillation or a one dimensional orbit.

14.3 Future Models

We could extend this model by adding the components s and t to our object (x, E) . Here, s is the amount of space the energy occupies and t is a unit of time t associated to the object. In this extended model, objects could be written as (x, E, s, t) . Based on this, our natural space could be written as $(x = 0, E = 0, s = 1, t = 1)$, an object with no speed, occupying a space and associated with a unit of time of 1. A photon might be interpreted as $(x = c/t, E = 0, s = 0, t = 1)$ and a neutrino as $(x = v_n/t, E = m_n, s = 0, t = 1)$ where v_n is the neutrino speed at some energy m_n . Based on these objects,

we could try to find a transformation force which account to many observed properties. Our object (x, E, s, t) could be represented as a rectangular prism such that the length is s , the width is t and height is E . Attached to the prism is also a vector of x pointing in some direction. If we find the right force transformation, we could modify the unit of time as seen in relativity, investigate consequences of an invariant volume and maybe discover inner mechanisms explaining the interaction between space, energy and time.

For example, if we take the sheet of space $(x = 0, E = 0, s = 1, t = 1)$ and assume that the area stays the same, stretching s to 2 would make $t = 1/2$. If we interpret t as being the time it took to cover the last distance x , then an object passing through this space will get the values $x = s = 2$ and $t = 1/2$, thus making the speed of the object $x/t = 4$ which can be understood as time dilatation.

Take again a sheet of space $(x = 0, E = 0, s = 1, t = 1)$ and assume that the area is an invariant. As time spends itself, then s becomes greater which could be interpreted as an expanding space. Time would need to be viewed as decreasing and not increasing, making time more like water evaporating.

Based on the assumption that photons are emitted in quanta, we are forced to conclude that there is a limit to what we can measure. This is also assuming that there are no other particles or techniques which can be used to have a lower limit. Since what we want to measure now takes the form of a rectangular prism, one way to look at Heisenberg's uncertainty principle is to take a small prism that has the sides of Planck's units and assume the volume is invariant. Thus, stretching a side of the prism will reduce the side of the others in a way which is similar to a reduction of Δx implies an augmentation of Δp_x . From this point of view, saying that our prism has an invariant volume, is another way to express the Heisenberg uncertainty principle.

15 Metascience

We now have the possibility to express each domain of science by using the same mathematical notation, this means that each science can directly benefit from the others. Structures found or studied in one science can be applied or studied in another domain. For example, we can try to see if a new mathematical structure also appears in biology, neuroscience or chemistry. The point of view that comes with the language of transformations might be

able to provide insights on the concept of emergence.

We are presently at a place where most researchers have to interact with different fields of science to improve their understanding of the world and create beneficial technologies. The language which was presented is only a few steps towards improving the free exchange of ideas and the creation of a large interactive open-access scientific database.

Languages such as the one presented combined with a large scientific database can prove to be fruitful and powerful tools. It is the sincere hope of the author that every person involved will take the responsibility of making ethical choices aimed at benefiting each and every living being.

16 Appendix

We will now give a few more definitions and tools that are useful to manipulate systems of transformations.

16.1 Open Transformations

Until now, in our transformations, the cause of the transformation was defined as being the same as the elements in the initial form. We will now introduce a notation saying that the transformation will be applied to any disconnected form it meets. The transformation

$$\boxed{X \triangleright \star}$$

is understood as replacing any symbol it meets by the form X . For example,

$$\boxed{X \triangleright \star} \rightarrow (a)$$

will reduce to (X) and

$$\boxed{X \triangleright \star} \rightarrow (b)$$

will also reduce to (X) .

We can also have more than one star in the cause of a transformation. For example,

$$\boxed{X, Y \triangleright \star, \star} \rightarrow (c, d)$$

will also reduce to (X, Y) .

We can also restrict the scope of the star symbol to a certain color by writing $\boxed{X \triangleright \star}$. If we apply this transformation three times to $\{a, b, a, c\}$, this will give $\{X, X, X, c\}$.

The star symbol can also be written in the result of a transformation. An example of this is

$$\text{Duplication} := \boxed{\star, \star \triangleright \star}$$

which permits the duplication of any form it meets. Applying this transformation to an initial form containing a rabbit will give us a set of two rabbits.

$$\boxed{\star, \star \triangleright \star} \rightarrow \{rabbit\} \Rightarrow \{rabbit, rabbit\}$$

Applying duplication two times to a set of three rabbits, will give a total of five rabbits and applying duplication three times to a set of three rabbits,

will give a total of six rabbits. If we want to apply the duplication to each rabbit in the initial set, we can use the sharp symbol and write the following.

$$\begin{aligned} \boxed{\star, \star}^\# &\rightarrow \{rabbit, rabbit, rabbit\} \\ &\Downarrow \\ &\{rabbit, rabbit, rabbit, rabbit, rabbit, rabbit\} \end{aligned}$$

In essence, we have multiplied the number of rabbits of the initial set by 2. Similarly, multiplication by 5 of a set of elements can be done with the transformation $\boxed{\star, \star, \star, \star, \star \triangleright \star}$.

Another example from chemistry, a transformation replacing any element having a bond with carbon by an hydrogen can be written as $\boxed{C - H \triangleright (C - \star)}$.

16.2 Negation

We now introduce the negation symbol \neg . When this symbol is written before a form, it means that we are referring to all that is not that form in a set. For example, take the set $Office := \{desk, computer, person, pen, phone\}$, we can replace by a beautiful meadow all that is not the person in the office by using the symbol \neg_{Office} .

$$\begin{aligned} [\text{beautiful meadow} \triangleright \neg_{Office}(person)] &\rightarrow \{desk, computer, person, pen, phone\} \\ &\Downarrow \\ &\{\text{beautiful meadow}, person\} \end{aligned}$$

We can also use the negation symbol in the effect of the transformation. Let $PollutedWater := \{Pollutants, water\}$, also denoted by PW , be the set containing water mixed with different pollutants. The following transformation can be interpreted as a water filter which filters out all pollutants.

$$\boxed{\neg_{PW}(Pollutants) \triangleright PollutedWater}$$

16.3 Initial set displacement

Until now the position or content of the initial set was fixed and did not change unless transformations were applied to it. We now want to show how we can have an initial form which is displaced after the application of a transformation. Take the string of letters *ababbbaab*, we can define a

transformation such that it is applied to the first three terms, then applied to the next three terms and then to the last three terms. We can use the ‘ \star ’ notation to allow us to move the initial form.

If we follow each transformation by the transformation

$$\boxed{\star\star\star(\star\star\star) \triangleright (\star\star\star) \star\star\star}$$

we will displace the initial form.

An example is given as follows.

$$\begin{array}{c} \boxed{\star\star\star(\star\star\star) \triangleright (\star\star\star) \star\star\star} \rightarrow \boxed{cc \triangleright ab} \rightarrow (aba)bbbaab \\ \downarrow \\ \boxed{\star\star\star(\star\star\star) \triangleright (\star\star\star) \star\star\star} \rightarrow (cca)bbbaab \\ \downarrow \\ cca(bbb)aab \end{array}$$

16.4 Series and parallel invariance

A collection of transformations applied in series to an initial form will often have different effects than if the transformations were applied in parallel. But depending on how the model was built, whether we apply the transformations in series or parallel, will have the same effect. Eventually, it would be interesting to find a precise characterisation of systems which are reduced in the same way whether the transformations are applied in series or in parallel. Here follows two examples of what we could call *parallel-series invariant* systems.

$$\boxed{B \triangleright b} \rightarrow \boxed{A \triangleright a} \rightarrow (ab) \asymp [\boxed{B \triangleright b}, \boxed{A \triangleright a}] \rightarrow (ab).$$

$$\boxed{cc \triangleright ba} \rightarrow \boxed{bba \triangleright aaa} \rightarrow (aaa) \asymp [\boxed{bba \triangleright aaa}, \boxed{cc \triangleright ba}] \rightarrow (a, a, a).$$

16.5 Typesetting

The paper was written in the typesetting language LaTeX. Most figures and diagrams were drawn in LaTeX Draw which generate PSTricks code that can be copied directly in the LaTeX document. The LaTeX animate package was used for the animations. Note that for the LaTeX code to compile well,

all the images needed to be of the Encapsulated PostScript (*.EPS*). For the animations to display well, the code needed to be compiled with XeLaTeX, but for rapid compilation the sequence LaTeX, DVI2PS and PS2PDF was used.

The transformations are written with the following LaTeX code.

$$\mathop{\fbox{$B \rhd A$}}^{\{\backslash, \infty S\}}$$

This code will display as follows.

$$\boxed{B \rhd A}^{\infty S}$$

Acknowledgement

I would like to thank Véronique Pagé and Nathalie Patenaude for their help and encouragement. I am also infinitely grateful to Anyen Rinpoche and Johanna Okker for their unfailing help and support. They are the ones who made the realization of this project possible.

References

- [1] Bezem, M., J. W. Klop, and R de Vrijer, eds. *Term rewriting systems*, Cambridge University Press, 2003.
- [2] Chomsky, N., “Three models for the description of language”, IRE Transactions on Information Theory, vol. 2, iss. 3, pp. 113-124, 1956.
- [3] Feynman, R. P., “The theory of positrons”, Physical Review 76, pp. 749-759, 1949.
- [4] Feynman, R. P., “Space-time approach to quantum electrodynamics”, Physical Review 76, pp. 769-789, 1949.
- [5] Prusinkiewicz, P., and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, New York, 1990.
- [6] Mac Lane, S., *Categories for the Working Mathematician*, Graduate Texts in Mathematics, Vol. 5, Springer-Verlag, New York, 2nd ed. 1978.
- [7] Penrose, R. , “Applications of negative dimensional tensors”, Applications of Combinatorial Mathematics, pp. 221-244, 1971.
- [8] Selinger, P., “A survey of graphical languages for monoidal categories”, Book chapter. In Bob Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics 813, Springer, pp.289-355, 2011.